UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

IMPLEMENTACIÓN DE AMAZON WEB SERVICES PARA ANÁLIS DE VIDEO

Rubén Pérez Martínez

Director: Òscar Amorós Huguet
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 27 de juny de 2019

· **Resumen**.

En este proyecto se presentarán diferentes formas de resolver el problema del procesado de video en tiempo real, en el momento de capturar las imágenes con la cámara. Tomaremos como ejemplo de caso de uso, un producto real en la industria. Dicho producto consiste en el caso más extremo, en 6 cámaras 4K a 60fps, un servidor en el *edge* conectado a dichas cámaras, y un software que genera un solo video en FullHD como resultado de la composición de dichas cámaras, y del análisis por visión por computador de solo 2 de ellas. Todo el cálculo, se puede hacer en el *edge* que es en el mismo lugar donde tienes las cámaras, y eso permite tener un buen ancho de banda, estable, controlado y barato, entre las cámaras y el servidor. Lo que permite obtener el video final con baja latencia respecto a la realidad, y alta calidad de imagen. Pero dependiendo de las necesidades de cálculo, el hardware puede ser muy caro (como el caso que presentamos) y su uso no es intensivo pues solo se usa en el momento de hacer el procesado. La otra opción es usar la nube, donde no te tienes que preocupar del mantenimiento del hardware, y tiene mucho software ya hecho, dándote recursos optimizados y centralizados. La parte negativa es que se necesitan conexiones estables con anchos de banda grandes para soportar cámaras 4K a 60 fps con buena calidad de *encoding* que suele implicar unos *bitrates* elevados.

En este proyecto exploraremos diferentes estrategias para dar solución al análisis de video, usando como plataforma en la nube Amazon Web Services. Presentaremos también una estrategia híbrida, en la que el análisis del video se realiza en la nube, y el uso de ese análisis, para generar la composición en FullHD se realiza en el *edge*. Consiguiendo así una reducción de los costes computacionales en el *edge* y de costes de conexión a la nube, ya que no enviaremos el video entero, si no solo algunos fotogramas de las dos cámaras implicadas en el análisis de visión por computador.

Resum

En aquest projecte es presentaran diferents formes de resoldre el problema del processament de vídeo en temps real, en el moment de capturar les imatges amb la càmera. Prendrem com a exemple de cas d'ús, un producte real en la indústria. Aquest producte consisteix en el cas més extrem, en 6 cambres 4K a 60 fps, un servidor en el *edge* connectat a aquestes càmeres, i un programari que genera un sol vídeo en FullHD com a resultat de la composició d'aquestes càmeres, i de l'anàlisi per visió per computador de sol 2 d'elles. Tot el càlcul, es pot fer en el *edge* que és en el mateix lloc on tens les càmeres, i això permet tenir una bona amplada de banda, estable, controlat i barat, entre les càmeres i el servidor. El que permet obtenir el vídeo final amb baixa latència respecte a la realitat, i alta qualitat d'imatge. Però depenent de les necessitats de càlcul, el maquinari pot ser molt car (com el cas que presentem) i el seu ús no és intensiu perquè només s'usa en el moment de fer el processament. L'altra opció és utilitzar el núvol, on no t'has de preocupar del manteniment del maquinari, i té molt programari ja fet, donant-te recursos optimitzats i centralitzats. La part negativa és que es necessiten connexions estables amb amplades de banda grans per a suportar càmeres 4K a 60 fps amb bona qualitat de *encoding* que sol implicar uns *bitrates* elevats.

En aquest projecte explorarem diferents estratègies per a donar solució a l'anàlisi de vídeo, usant com a plataforma en el núvol Amazon Web Services. Presentarem també una estratègia híbrida, en la qual l'anàlisi del vídeo es realitza en el núvol, i l'ús d'aquest anàlisi, per a generar la composició en FullHD es realitza en el *edge*. Aconseguint així una reducció dels costos computacionals en el *edge* i de costos de connexió al núvol, ja que no enviarem el vídeo sencer, si no només alguns fotogrames de les dues càmeres implicades en l'anàlisi de visió per computador.

Abstract

This project will present different ways to solve the problem of video processing in real time, at the time of capturing the images with the camera. We will take as an example of use case, a real product in the industry. This product consists in the most extreme case, in 6 cameras 4K at 60fps, a server at the edge connected to these cameras, and a software that generates a single FullHD video as a result of the composition of these cameras, and computer vision analysis of only 2 of them. All the calculation can be done at the edge that is in the same place where you have the cameras, and that allows to have a good bandwidth, stable, controlled and cheap, between the cameras and the server. This allows you to obtain the final video with low latency with respect to reality, and high image quality. But depending on the needs of calculation, the hardware can be very expensive (as the case that we present) and its use is not intensive because it is only used in the moment of doing the processing. The other option is to use the cloud, where you do not have to worry about hardware maintenance, and has a lot of software already done, giving you optimized and centralized resources. The negative side is that you need stable connections with large bandwidths to support 4K cameras at 60 fps with good encoding quality that usually involves high bitrates.

In this project we will explore different strategies to solve video analysis, using Amazon Web Services as a cloud platform. We will also present a hybrid strategy, in which video analysis is done in the cloud, and the use of that analysis to generate the FullHD composition is done at the edge. This way we achieve a reduction of computational costs at the edge and of connection costs to the cloud, since we will not send the whole video, but only some frames of the two cameras involved in the computer vision analysis.

Índice

Índice Figuras

1- Introducción y motivación

El análisis de video es hoy en día un campo muy importante de la ingeniería informática. Podemos ver ejemplos en todos los lugares, uno de los más importantes sería el de los coches autónomos, también en los sistemas de video vigilancia, que tanto los EEUU como china están investigando para las llamadas futuras ciudades inteligentes. Su uso también comienza a extenderse a eventos deportivos, donde gracias a las nuevas tecnologías, se puede hacer por ejemplo que las cámaras sigan la pelota, reduciendo así costes en personal que controla las cámaras.
Las plataformas de servicios en la nube se han convertido en uno de los mercados de las tecnologías más importantes de los últimos años. Todas las grandes empresas están invirtiendo cuantiosas cantidades de dinero para ponerse a la vanguardia en este campo. Microsoft con Azure, Google con Google Cloud y en el caso que nos ocupa Amazon con Amazon Web Services son los más importantes en la actualidad.
Así mismo, hay ciertos casos de uso (coches autónomos y robótica) que, por la cantidad de datos capturados, y la necesidad de inmediatez en la obtención de resultados, no se pueden llevar a la nube, y que se pueda en un futuro es cuestionable por lo menos a corto plazo. Para estos problemas existe lo que recientemente se ha llamado Edge computing. Lo que significa procesar los datos allí donde se generan.

Recientemente, empresas conocidas por sus servicios cloud como Amazon, están empezando a ofrecer soluciones Edge, en combinación con sus servicios en la nube. Es en esta nueva modalidad de diseño de soluciones, donde radica el interés de este trabajo que explora los costes del uso exclusivo de la nube, y busca la mejor combinación entre el edge y la nube para minimizar costes.

2- Contexto

Como ya hemos comentado en el resumen, existe un caso de uso real en la industria, donde se requiere capturar varias cámaras de alta resolución (hasta 6 cámaras 4K), a *framerates* también altos (60fps). El software de análisis solo tiene que analizar algunos fotogramas de algunas (un par) de esas cámaras, y otra parte del software genera un video como resultado, que es la combinación de todas las cámaras. Ese video resultante ha de mantener al máximo la calidad de imagen original, con lo que si hay que comprimir el video para mandarlo al servidor que va a producir el video de salida, es necesario que el *bitrate* del video sea elevado (alrededor de unos 150Mbps).

Para este trabajo, por simplificar el problema y reducir los costes, usaremos una sola cámara de menor *framerate* y resolución.

3- Objetivos

Respecto al caso de uso descrito, en este trabajo nos centraremos en cómo solucionar la parte del análisis del video, ya que es la parte de código más variable, y que requiere más capacidad de cálculo. La otra parte mencionada, en la que se combinan las diferentes cámaras en un solo video, es una pieza de software que se puede ejecutar en cualquier parte porque requiere menos rendimiento, y solo formará parte del análisis de soluciones, en el punto de definir dónde se va a ejecutar.
Para cumplir con el objetivo de encontrar soluciones *cloud*, he analizado en profundidad dos partes importantes de las capacidades de Amazon Web Services. Por un lado, las capacidades y costes de ancho de banda para enviar videos en *streaming*, y las capacidades de análisis de video y como poder aplicarlas a dar solución al problema de analizar video en tiempo real. Para eso, he desarrollado el software o utilizo ejemplos que proporciona la misma Amazon o terceros para poder encontrar sus puntos débiles y fuertes. Los objetivos se pueden desglosar en:

- Entender cómo funciona Amazon Web Services, las ventajas y desventajas que conlleva el trabajar en la nube.
- Diseñar o aplicar arquitecturas ya hechas que resuelvan casos de uso donde el análisis de video sea necesario.
- Aplicar, desarrollar, mejorar estas arquitecturas con diferentes filosofías.
- Probarlas y comprobar rendimiento.
- Analizar costes.

4-Planificación

Posición	Fecha de inicio	Fecha de finalización	Hito o actividad
1	1/2/19	26/2/19	Formación
2	1/3/19	30/3/19	Análisis de requerimientos
3	1/4/19	25/5/19	Programación
4	26/4/19	25/5/19	Pruebas
5	26/5/19	26/6/19	Memoria

Figura 1: Calendario de planificación TFG.

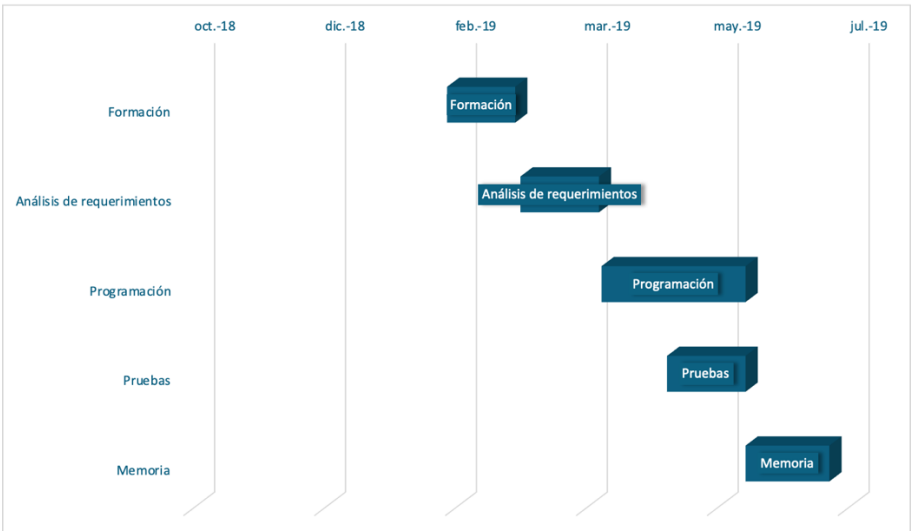


Figura 2: Diagrama de Gantt planificación.

5- Análisis

Ante la necesidad de trabajar con video y analizar los fotogramas a través de AWS es necesario una forma de enviar estos hacia sus servidores. Existen dos formas distintas de hacer esto. Una que es fotograma a fotograma en intervalos escogidos para evitar al máximo saturar el trafico de datos y otra con un *streaming* continuo de video. Amazon proporciona varias formas que pasaré a explicar.

5.1 Envío de fotogramas, datos y video con AWS.

Amazon Kinesis Data Stream proporciona la capacidad de enviar datos a donde nosotros queramos dentro de AWS, lo usaremos para poder enviar los fotogramas de forma individual en intervalos a determinar en función de las necesidades directamente a una función Lambda, pero también sirve para la recopilación, el procesamiento y el análisis de datos de *streaming* en tiempo real. Es auto escalable y permite procesar y analizar datos a medida que se reciben y responder instantáneamente en vez de tener que esperar a que los datos se recopilen antes de que el procesamiento pueda comenzar.
KDS puede registrar de manera continua gigabytes de datos por segundo de cientos de miles de orígenes, como transmisiones de clics de sitios web, transmisiones de eventos de bases de datos, transacciones financieras, fuentes de redes sociales, registros de TI y eventos de seguimiento de ubicaciones. Los datos recopilados se encuentran disponibles en milisegundos para posibilitar los casos de uso de análisis en tiempo real, como paneles en tiempo real, detección de anomalías en tiempo real y precios dinámicos, entre otros. [1]

En el caso de transmitir video se necesita una herramienta pensada para ello, por eso Amazon proporciona **Amazon Kinesis Video Streams** [2], este facilita la transmisión segura de videos desde dispositivos conectados a AWS para tareas de análisis, aprendizaje automático (ML), reproducción y otros procesos. Kinesis Video Streams aprovisiona automáticamente y escala de manera elástica toda la infraestructura necesaria para incorporar datos de las transmisiones de vídeo de millones de dispositivos. También almacena, cifra e indexa de forma duradera datos de videos en sus transmisiones, y permite obtener acceso a sus datos mediante API fáciles de usar. Kinesis Video Streams le permite reproducir videos para visualizaciones en directo y bajo demanda, y crear rápidamente aplicaciones que aprovechan la visión artificial y el análisis de videos a través de la integración con Amazon Rekognition Video, y bibliotecas para marcos de ML, como Apache MxNet, TensorFlow y OpenCV.
El limite de ancho de banda para subir video de un *stream* es de 100 Mbps (12.5 MB/s) con lo que sería posible hacer un *stream* de 4K 30 fps. [3]

Para poder enviar tanto el fotograma individual como el video en directo necesitaremos un productor que envíe estos datos capturados por la cámara.
En el caso de los fotogramas por separado se usará OpenCV ya que permite capturar fotograma por fotograma de forma sencilla.
Se escoge OpenCV pues es de código libre y se encuentra mucha literatura en internet para facilitar su programación en python.
En el caso del productor del *streaming* de video se usará un SDK proporcionado por Amazon que usa GStreamer [4].

Una vez enviados estos fotogramas se necesita definir que servicios los van a consumir, los consumidores son aplicaciones personalizadas que consumen y procesan los fotogramas o el *streaming* de video que es enviado a través de Kineisis Data Stream o Kinesis Video Stream en tiempo real o después de que estos se almacenen. Estas aplicaciones pueden ser de la propia Amazon como Rekognition, pueden estar alojadas en una instancia EC2 como DeepStream o pueden ser proveedores de análisis de video de terceros.

5.2 Amazon Rekognition.

Amazon Rekognition es el servicio de Amazon que se utilizará en varias de las soluciones hechas en este trabajo. Este facilita la incorporación del análisis de imágenes y videos a nuestras aplicaciones. Tan solo tenemos que suministrar una imagen o video a la API de Rekognition y el servicio identificará objetos, personas, texto, escenas y actividades, además de detectar contenido inapropiado. Amazon Rekognition también ofrece análisis y reconocimiento facial de alta precisión en las imágenes y los videos que se suministren. Esto nos será útil para el proyecto de detección de caras que explicaremos más adelante. Puede detectar, analizar y comparar rostros para una amplia variedad de casos de uso de verificación de usuarios, contabilización de personas y seguridad pública.
Amazon Rekognition está basado en la misma tecnología de aprendizaje profundo sólida y de alta escalabilidad que desarrollaron los científicos de visión artificial de Amazon para analizar miles de millones de imágenes y videos diariamente. No es necesario contar con experiencia en aprendizaje automático para utilizarla. Amazon Rekognition es una API simple y fácil de usar que puede analizar rápidamente cualquier archivo de video o imagen almacenado en Amazon S3. Amazon Rekognition siempre está aprendiendo de los datos nuevos. Amazon asegura que agregan nuevas características de reconocimiento facial y etiquetas al servicio continuamente. [5]

5.2.1 Diferentes tipos de detección y reconocimiento con Amazon Rekognition

Amazon Rekognition tiene diferentes APIs [6] con las que hacer diferentes funciones dentro de la detección y reconocimiento, además de estar diferenciadas para detección y reconocimiento de Imágenes y para Video ahora pasaré a explicar los diferentes tipos y las funciones que se pueden hacer con cada una.

5.2.1.1 Etiquetas (labels)

Esta función proporciona información para detectar etiquetas en imágenes y videos.
Una etiqueta o marca es una escena, concepto o objeto encontrado en un video o en una imagen. Las etiquetas también devuelven su ubicación exacta (*BoundingBox*) de los objetos detectados como también el porcentaje de confianza de que ese objeto detectado es el que pone en la etiqueta. No se devolverá *BoudingBox* en el caso de objetos menos comunes.
Tanto Rekognition Image como Rekognition Video usan una taxonomía jerárquica de etiquetas antecesoras para categorizar las etiquetas. Esto quiere decir que si hay una persona cruzando la calle podría detectarse como *Pedestrian* (Peatón) y la etiqueta principal de esta sería *Person* (Persona). Se devolverían las dos etiquetas en la respuesta.

Las etiquetas nos pueden dar información de los siguientes elementos:

- Objetos (flor, árbol, teléfono, arma, gafas…)
- Eventos (partido de futbol, música en directo, boda…)
- Conceptos (naturaleza, ciudad, paisaje…)
- Actividades (chutar pelota, salir de un vehículo…)

La función de detectar etiquetas de Amazon Rekognition está disponible tanto para videos como para imágenes, pero las actividades solo están disponibles en los videos.
La API para detectar etiquetas:

- "DetectLabels" en el caso de las imágenes.
- "StartLabelDetection" en el caso de los videos.

Figura 3: ejemplo de posición de etiquetas.

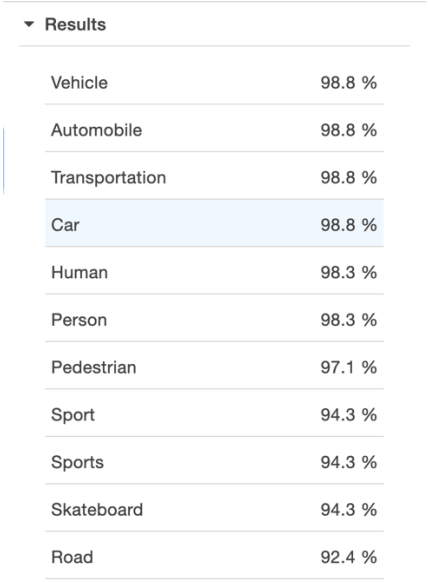


Figura 4: etiquetas encontradas y sus porcentajes de confianza.

5.2.1.2 Detección de caras

Con Rekognition se pueden detectar caras en imágenes y videos almacenados o en streaming con Amazon Kinesis Video Streams. Tiene la capacidad de:

- Un sistema de detección facial donde se determina la presencia, la ubicación, la escala y aproximadamente la orientación de los rostros que aparecen en la imagen o el fotograma de vídeo. Puede detectar caras independientemente de atributos como la edad si tiene barba o el género.
- Un sistema de reconocimiento facial donde a partir de una imagen con un rostro nos indica si ese rostro se repite en otras imágenes de una base de datos especificada independientemente de su expresión, vello facial y edad.

Podemos obtener información como:

- Cuadro delimitador (*BoudingBox*): Las coordenadas del cuadro delimitador que rodea el rostro.
- Confianza: Grado de confianza de que el cuadro delimitador contiene una cara.

- Referencias faciales: Una matriz de referencias faciales. Para cada referencia (como el ojo izquierdo, el ojo derecho y la boca), la respuesta proporciona las coordenadas x, y.
- Atributos faciales: como el sexo, si el rostro tiene barba. Para cada atributo, la respuesta proporciona un valor. El valor puede ser de diferentes tipos, como un tipo booleano (si una persona lleva gafas), una cadena (si la persona es un hombre o una mujer), etc. Además, para la mayoría de los atributos la respuesta proporciona también una confianza en el valor detectado para el atributo.
- Calidad – Describe el brillo y la nitidez del rostro.
- Postura – Describe la rotación del rostro dentro de la imagen.
- Emociones – Un conjunto de emociones con confianza en el análisis. Como triste, enfadado, alegre…

Podemos también comparar imagen con otra para encontrar coincidencias de rostros, a través de una imagen origen compararla con una de destino. La respuesta devolverá la posición donde se encuentra la cara similar y su porcentaje de confianza.

La API para iniciar la detección de caras:

- "DetectFaces" en imágenes
- "StartFaceDetection" en video
- "CompareFaces" en imágenes

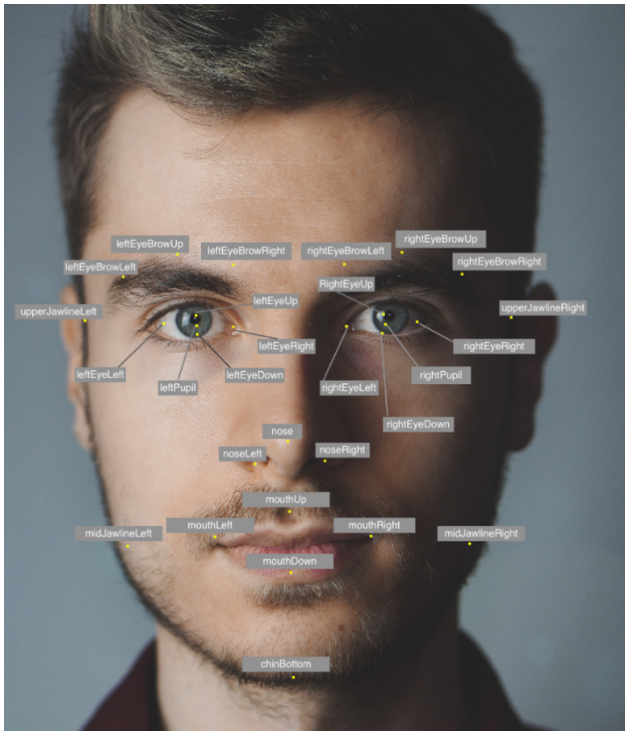


Figura 5: imagen que muestra la ubicación relativa de las referencias faciales (*Landmarks*) devueltas [7]

5.2.1.3 Detección de caras de entre una colección

Podemos almacenar información sobre rostros en contenedores conocidos como colecciones. Con esto podemos usar la información de las caras almacenada en una colección para buscar caras conocidas en imágenes, videos almacenados y en videos por *streaming*.

5.2.1.4 Recorridos de las personas

Podemos utilizar Amazon Rekognition para hacer un seguimiento de los recorridos de las personas detectadas en un video almacenado. Como es lógico esta opción no esta para Rekognition de imágenes, pero tampoco lo está para video en *streaming* únicamente podremos hacer el seguimiento de personas a través de videos almacenados.
Está función no serviría para por ejemplo hacer también un seguimiento de un balón de futbol en un partido o seguir el recorrido de coches.

La información que nos devuelve Rekognition en este caso es:

- Información de persona: nos da información de la hora en que ha sido detectada la persona (*Timestamps*), la posición de la persona detectada en el fotograma (*BoundingBox*), la confianza que tiene Rekognition de que es una persona (*Confidence*).
- Información de la paginación: en el caso de haber puesto un limite de personas a detectar y este es superado en un fotograma se indicará un *NextToken* para ver más resultados.
- Índice: indicador único de la persona en todo el video.
- Información del video: aquí se indica en cada pagina devuelta la información del video, como el formato, la duración total y la resolución.

StartPersonTracking será el nombre de la API de detectar personas en videos almacenados.

5.2.1.5 Detectar Famosos

Podemos reconocer miles de famosos en imágenes y videos almacenados y en este caso tampoco en video *streaming*.
La información que devuelve:

- Nombre del famoso
- Donde se encuentra el rostro del famoso (*BoundingBox*)
- Referencias faciales
- Postura del famoso
- Información de seguimiento de famosos a medida que aparecen en un video almacenado.
- Información adicional del famoso

Para reconocer famosos las APIs son:

- “RecognizeCelebrities” en imágenes

- “StartCelebrityRecognition” en videos almacenados

5.2.1.6 Detección de video no seguro

Podemos determinar si una imagen o un video incluye contenido para adultos explicito o insinuante con la API DetectModerationLabels para imágenes y StartContentModeration y GetContentModeration para video. Estas APIs devolverán en que categoría se encuentran de detectarse algún nivel de contenido adulto en las fotos dentro de un porcentaje de confianza.

5.2.1.7 Detección de texto

Podemos detectar texto en imágenes y convertirlo en texto legible por una máquina. DetectText es el nombre de la API para realizar la detección de texto.

Limites de Amazon Rekognition Imágenes:

- Una imagen almacenada como objeto de Amazon S3 tiene que tener un tamaño máximo de 15MB
- El tamaño mínimo en pixeles que tiene que tener una imagen es de 80 x 80 pixeles.
- Para la detección de rostros en una imagen de 1920 x 1080 debe ocupar 40 x 40 pixeles. Si la imagen es superior a 1920 x 1080 pixeles el tamaño mínimo del rostro debe ser proporcionalmente mayor.
- El tamaño máximo de imagen como bytes sin procesar pasados como parámetro a una API es de 5 MB.
- Los formatos admitidos de imágenes de entrada para las diferentes APIs de Rekognition son PNG y JPEG.
- Se pueden almacenar en una única colección de rostros un máximo de 20 millones.
- El número máximo de ostros coincidentes devueltos por la API de búsqueda es de 4096.

Límites de Amazon Rekognition Video:

- Máximo de 8 GB de tamaño por video a analizar.
- Máximo de 2 horas de duración por video a analizar.
- Máximo de 20 trabajos simultáneos por cuenta.
- En un procesador de *streaming* (StreamProcessor) tanto la transmisión de entrada de Kinesis Video como la transmisión de salida Kinesis Data no pueden estar asociadas a más de 1 procesador de *streaming*. Tampoco estas pueden estar asociadas a otros procesos de Amazon Rekognition.

Nota: algunos de estos límites pueden ser modificados por medio del soporte técnico de AWS con un formulario.

5.3 Herramientas usadas de AWS en las implementaciones

Con **Amazon Elastic Compute Cloud** (**EC2**) [8]podemos crear instancias en la nube con muchas combinaciones de Hardware modificables y seguras. A través de una sencilla interfaz EC2 permite obtener y configurar de forma muy rápida una instancia. En el caso de la solución de procesado de Imagen ejecutada por DeepStream es necesaria una instancia con una tarjeta gráfica de Nvidia, se puede seleccionar una a través de diferentes opciones o con plantillas ya hechas. En este caso solo se paga por el tiempo utilizando la instancia y este será más o menos elevado en función del Hardware escogido. La propia Nvidia proporciona una imagen Docker con todo lo necesario para ejecutarse.

Para la construcción del proyecto "Serverless" necesitaremos que cada vez que se envíe un fotograma a través de Kinesis algo lo gestione, en este caso Amazon proporciona **Lambda**, es un servicio de informática sin servidor que ejecuta código en respuesta a eventos y administra automáticamente los recursos informáticos subyacentes por lo tanto lo configuraremos para que cada vez que reciba un fotograma lo envíe a Rekognition. Se puede usar AWS Lambda para ampliar la funcionalidad de otros productos de AWS con lógica personalizada o bien crear servicios *back-end* propios que funcionen con el nivel de seguridad, rendimiento y escala de AWS. AWS Lambda puede ejecutar código automáticamente en respuesta a varios eventos, como solicitudes HTTP a través de Amazon API Gateway, modificaciones realizadas en objetos en buckets de Amazon S3, actualizaciones de tablas en Amazon DynamoDB y transiciones de estado en AWS Step Functions. [9]

Cada vez que Rekognition devuelve las etiquetas de cada fotograma que analiza y alguna de ellas esté entre la lista de objetos a alertar y queremos que nos envíe un correo o en el caso de este trabajo un SMS usaremos **Amazon Simple Notification Service (SNS)**, es un servicio de mensajería de publicación/suscripción completamente administrada, de alta disponibilidad, seguro y con durabilidad que permite desacoplar microservicios, sistemas distribuidos y aplicaciones sin servidor. Amazon SNS proporciona temas para la mensajería de alto rendimiento, basada en inserción y para muchos destinatarios. Mediante el uso de temas de Amazon SNS, los sistemas de publicadores pueden distribuir los mensajes a un gran número de puntos de enlaces de suscriptores para el procesamiento paralelo, incluidas las colas de Amazon SQS, las funciones de AWS Lambda y los *webhooks* HTTP/S. Además, SNS se puede usar para distribuir notificaciones a usuarios finales mediante notificaciones de inserción móviles, SMS y correo electrónico. [10]

Amazon Simple Queue Service (SQS) es un servicio de colas de mensajes completamente administrado que permite desacoplar y ajustar la escala de microservicios, sistemas distribuidos y aplicaciones sin servidor. SQS elimina la complejidad y los gastos generales asociados con la gestión y el funcionamiento del middleware orientado a mensajes, y permite a los desarrolladores centrarse en la diferenciación del trabajo. Con SQS, puede enviar, almacenar y recibir mensajes entre componentes de software de cualquier volumen, sin pérdida de mensajes ni la necesidad de que otros servicios estén disponibles.[11]

Las principales diferencias entre SNS y SQS son:
Amazon SNS permite a las aplicaciones enviar mensajes en los que el tiempo es esencia a varios suscriptores a través del mecanismo "*push*", lo que elimina la necesidad de comprobar o "sondear" de forma periódica en busca de actualizaciones. Amazon SQS es un servicio de cola

de mensajes que utilizan aplicaciones distribuidas para intercambiar mensajes a través del modelo de sondeo y se puede utilizar para desacoplar el envío y la recepción de componentes. [12]

Almacenar video rompe con la idea del análisis de video en tiempo real, pero no podemos descartarla del todo para otras posibilidades como la de hacer un *streaming* una vez terminado el análisis perdiendo así el componente "*live*" pero ahorrando dinero.
Es necesario tenerla en cuenta pues únicamente se pueden hacer según que análisis de video con Rekognition si estos están almacenados. Como el *person tracking*, LabelDetection, etc.
Para almacenar los fotogramas o los videos a Analizar utilizaremos **Amazon Simple Storage Service** (**Amazon S3**) [13], es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes en el sector. Esto significa que clientes de todos los tamaños y sectores pueden utilizarlo para almacenar y proteger cualquier cantidad de datos para diversos casos de uso, como sitios web, aplicaciones móviles, procesos de copia de seguridad y restauración, operaciones de archivado, aplicaciones empresariales, dispositivos IoT y análisis de *big data*. El tamaño máximo para un video almacenado es de 8GB.

Después de analizar cada fotograma Rekognition devuelve un json con toda la información resultante, si se quiere almacenar esta información asociada a un fotograma la mejor opción es **Amazon DynamoDB** es una base de datos de claves-valor y documentos que ofrece rendimiento en milisegundos de un solo dígito a cualquier escala. Se trata de una base de datos multirregión y multimaestro completamente administrada, con seguridad integrada, copia de seguridad y restauración, y almacenamiento de caché en memoria para aplicaciones a escala de Internet. En la tabla guardaremos los datos que más nos interesen del json devuelto junto a una *key* que está asociada al fotograma del que trata la información. [14]

Para poder ver los resultados del análisis del fotograma a través de una web usaremos otra función lambda que esta vez como evento a esperar sea un HttpMethod GET y que esta se encargue de servir la información guardada, el fotograma guardado en S3 y sus características que están en la tabla DynamoDB para hacer estas peticiones GET y hacer de intermediario con la web se utiliza **Amazon API Gateway** que es un servicio completamente administrado que facilita a los desarrolladores la creación, la publicación, el mantenimiento, el monitoreo y la protección de API a cualquier escala. Se puede crear API REST y API WebSocket que actúen como "puerta delantera" para que las aplicaciones obtengan acceso a datos, lógica de negocio o funcionalidades desde sus servicios *backend*, tales como cargas de trabajo ejecutadas en Amazon Elastic Compute Cloud (Amazon EC2), código ejecutado en AWS Lambda, cualquier aplicación web o aplicaciones de comunicación en tiempo real.
API Gateway no requiere pagos mínimos ni costos iniciales. Solo se paga por las llamadas a las API que se reciben y por la cantidad de datos salientes transferidos. [15]

AWS CloudFormation suministra un lenguaje común para describir y aprovisionar todos los recursos de la infraestructura en el entorno en la nube. CloudFormation permite usar un archivo de texto simple para modelar y aprovisionar, de una manera segura y automatizada, todos los recursos necesarios para las aplicaciones en todas las regiones y cuentas. El archivo funciona como la única fuente de información para el entorno en la nube. [16]

5.4 Envío de video en tiempo real sin análisis.

Otra forma de poder enviar video con Amazon es **Amazon Elemental MediaLive** [17], es un servicio de video en tiempo real que nos permite crear outputs en directo para *streming* y emisión en directo a muchos dispositivos.
Está pensado para poder transformar contenido de video en directo desde un formato y empaquetarlo a otros formatos y empaquetados para que pueda ser visualizado en smartphones y televisores. MediaLive puede tener como *input* y *output streams* de videos de diferentes tipos usando los estándares de la industria. La principal diferencia con Kinesis Video Stream es que MediaLive está optimizado para la emisión a muchos usuarios en una calidad de video alta, por el contrario, KVS esta optimizado para aplicaciones de uso de esas imágenes como Amazon Rekognition análisis de reconocimiento de caras, almacenado de video, o el uso de APIs en tiempo real.

5.5 Alternativa a Rekognition

5.5.1 Amazon SageMaker

Como alternativa a Rekognition se puede utilizar Amazon **SageMaker** [18], nos permite:

- Preparar y recopilar los datos de entrenamiento. Etiquetado de datos y blocs de notas prediseñados para los problemas comunes. Utilizaremos uno de estos para la detección de objetos en un video.
- Configurar y administrar los entornos para el entrenamiento con un solo clic en la infraestructura de más alto rendimiento.
- Implementar con un solo clic el modelo en producción.

5.5.2 Nvidia DeepStream

Una alternativa potente fuera de AWS sería **Nvidia DeepStream** [19] esta puede ser instalada en el *edge* o en una instancia EC2. Este software no ha sido utilizado en este trabajo, pero he creído importante buscar una tercera forma de hacer análisis de video con un software de terceros.
DeepStream es un SDK de análisis de *streaming* de video de propósito general con el podemos desarrollar potentes aplicaciones de análisis de video usando NVIDIA Jetson o plataformas Nvidia Tesla.

DeepStream SDK usa GStreamer que es un fotogramawork *open-source* de entrega de video de alto rendimiento con baja latencia. GStreamer es una librería para la construcción de grafos de componentes de manejo de medios de comunicación. Puedes construir aplicaciones que van desde un simple *streaming* de video y reproducirlos hacia grafos para su procesamiento mediante la IA. En este caso DeepStream.

DeepStream SDK de NVIDIA ofrece un completo kit de herramientas de análisis de transmisión para conocer la situación a través de la visión por computadora, el análisis de video inteligente y el procesamiento de sensores múltiples. El marco de la aplicación DeepStream cuenta con bloques de construcción acelerados por hardware que llevan las redes neuronales profundas y otras tareas de procesamiento complejas a una línea de procesamiento de flujo. Puedes

centrarte en crear redes centrales de aprendizaje profundo en lugar de diseñar soluciones de principio a fin desde cero.

DeepStream SDK es parte de CUDA-X Metropolis; una completa plataforma de software para crear soluciones de extremo a extremo con visión artificial, análisis de video inteligente (IVA) y procesamiento de sensores múltiples. CUDA-X Metropolis incluye DeepStream SDK para desarrollo de aplicaciones, Transfer Learning Toolkit para capacitación, TensorRT para inferencia acelerada, Video Codec SDK para aceleración de hardware de multimedia y más.

También el poder ejecutar DeepStream a través de una instancia Amazon EC2.

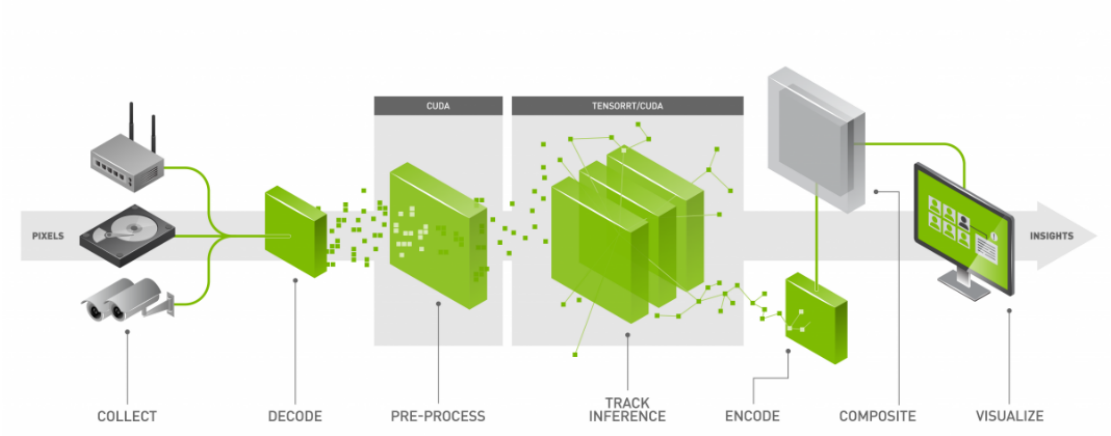


Figura 6: DeepStream

5.6 Seguridad en AWS

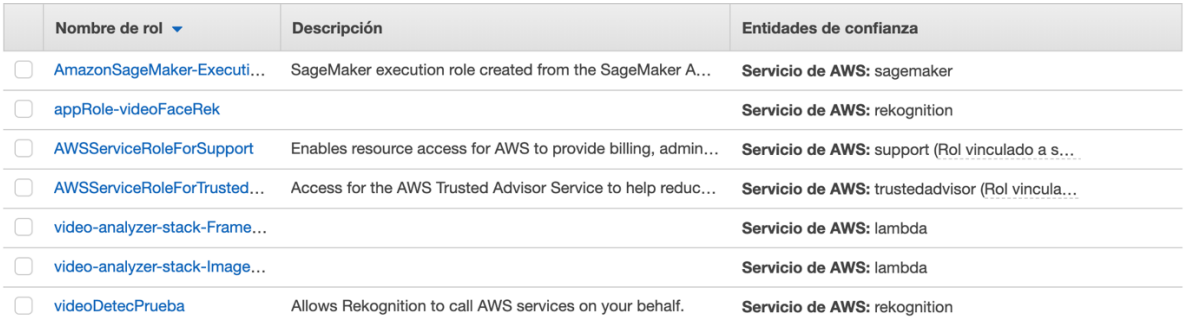
Para poder gestionar quien entra y a que tiene acceso y de que modo Amazon proporciona el servicio Amazon IAM [41]con el que poder crear:

- Usuarios
- Grupos
- Roles
- Políticas

Un usuario es una identidad única reconocida por los por los servicios y aplicaciones de AWS. Cada usuario dispone de un nombre único y puede identificarse utilizando las credenciales. A un usuario se le puede dar acceso a algún servicio o a todos de AWS bajo el control y la responsabilidad de la cuenta de AWS para la que se define.

Un grupo es un conjunto de usuarios de IAM. Se puede añadir usuarios a un grupo o eliminarlos del grupo. Un mismo usuario puede pertenecer a carios grupos. Por el contrario, los grupos no pueden pertenecer a otros grupos. Se pueden conceder permisos a los grupos mediante las políticas de control de acceso. Los grupos no tienen credenciales de seguridad y no pueden obtener acceso de forma directa a los servicios web.

Los roles o funciones de IAM son una forma segura de conceder permisos a entidades de confianza como usuarios. Una función de IAM es una entidad de IAM que define un conjunto de permisos para realizar solicitudes de servicio de AWS. Las funciones o roles no están asociadas con ningún grupo ni usuario especifico, como usuarios de IAM, aplicaciones o servicios de AWS como EC2. En el caso del proyecto será el usuario principal con el que se trabajará a través de AWS CLI. Por ejemplo en python se puede crear un rol con la instrucción create_role() una vez creado el rol se le asocia una política donde estarán los permisos que puede tener ese rol con put_rol_policy().



Nombre de rol	Descripción	Entidades de confianza
AmazonSageMaker-Executi...	SageMaker execution role created from the SageMaker A...	**Servicio de AWS:** sagemaker
appRole-videoFaceRek		**Servicio de AWS:** rekognition
AWSServiceRoleForSupport	Enables resource access for AWS to provide billing, admin...	**Servicio de AWS:** support (Rol vinculado a s...
AWSServiceRoleForTrusted...	Access for the AWS Trusted Advisor Service to help reduc...	**Servicio de AWS:** trustedadvisor (Rol vincula...
video-analyzer-stack-Frame...		**Servicio de AWS:** lambda
video-analyzer-stack-Image...		**Servicio de AWS:** lambda
videoDetecPrueba	Allows Rekognition to call AWS services on your behalf.	**Servicio de AWS:** rekognition

Figura 7: Lista de roles utilizados para este proyecto

Las políticas administrativas son recursos de IAM que expresan permisos mediante el lenguaje de políticas de IAM. Se pueden crear, editar, seleccionar de una lista muy extensa ya predefinida y administrarlas independientemente de los usuarios, grupos y funciones de IAM a los que están adjuntas. Una vez se asocia una política a un usuario, grupo o rol si esta se modifica se aplicará automáticamente a todas las entidades adjuntas a ella.

Para poder trabajar a través de python y la librería Boto3 con los servicios de AWS es necesario instalar AWS Command Line Interface (AWS CLI).
AWS CLI es una herramienta *open source* que te permite interactuar con los servicios de AWS usando comandos en tu la propia línea de comandos con una mínima configuración. Proporciona acceso directo a las APIs públicas de los servicios de AWS y así poder desarrollar Shell scripts para trabajar con los recursos. También da acceso a poder usar las SDKs de AWS como boto3. [20]

Boto es un SDK de AWS para python. Permite a los desarrolladores crear, configurar, y manejar los servicios de AWS. [21]

Una vez instalado AWS CLI hay que introducir las credenciales del usuario con el que quiere trabajar.
Con la llamada “aws configure” en un Shell podremos introducir los siguientes campos:

- AWS Access Key ID
- AWS Secret Access Key
- Default region name
- Default output format

Una vez introducidos quedará guardado como perfil a usar cada vez que se usa AWS CLI.

Access Key y Secret Access Key:
Estos dos son las credenciales a AWS. Estas están asociadas con una AWS Identity and Access Management (IAM) usuario o rol que determina los permisos que tienes.

Default region name sirve para identificar la región de AWS en la que por defecto se enviarán las peticiones por defecto. Lo más lógico es poner una región que esté lo más próxima posible a la tuya. Una vez especificada la región todas las peticiones irán a esta si no se especifica lo contrario.

5.7 Características cuenta gratuita AWS

Para poder acceder a todos los servicios de Amazon Web Service es necesario crear una cuenta. Amazon proporciona para los primeros 12 meses acceso gratuito a los diferentes servicios con diferentes niveles de acceso a estos.

A continuación, mostraré de los servicios que necesitamos para llevar a cabo el proyecto el acceso que permite Amazon de forma gratuita [22] de estos, todas son al mes:

- Amazon EC2: 750 Horas de acceso gratuito a instancias con Windows, Linux, RHEL, o SLES. Con arquitectura t2.micro
- Amazon S3: al mes 5GB de almacenamiento, 20000 Get Requests, 2000 Put Requests.
- DynamoDB: 25GB de almacenamiento.
- Lambda: 1000000 de consultas, hasta 3,2 Millones de segundos de tiempo de computo.
- Api Gateway: 1 Millón de llamadas a la API.
- Rekognition: 5000 análisis de imágenes.

6- Implementación

6.1 Hardware empleado en las implementaciones

Antes de la implementación necesitamos explicar que hardware se ha usado para todas las soluciones probadas.

Necesitaremos una cámara para capturar el video ya sea para solamente quedarnos con fotogramas sueltos o producir un *stream* de video.
Para todo el proyecto se ha utilizado la cámara FaceTime HD del portátil MacBook Pro de 2015 cuya máxima resolución es de 1280x720 (720p) y un máximo de 30 fotogramas por segundo.

En cuanto al Hardware utilizado como *edge* para ejecutar y procesar los datos de salida de la nube se usa el mismo MacBook Pro 2015 las especificaciones de este son:

- Intel Core i5 de doble núcleo a 2,7 GHz (Turbo Boost de hasta 3,1 GHz) con 3 MB de caché de nivel 3 compartida.
- 8 GB de memoria LPDDR3 integrada a 1.866 MHz.
- 256 GB de almacenamiento flash PCIe.

- Intel Iris Graphics 6100.
- Conexión inalámbrica Wi-Fi 802.11ac; compatible con las normas 802.11a/b/g/n del IEEE.
- macOS Mojave.

Aquí trataremos la implementación de las diferentes aplicaciones de análisis de video con Amazon Web Services.

6.2- Implementación "Serverless Solution for Video Fotograma Analysis and Alerting".

Ejemplo proporcionado por AWS y su autor es, Moataz Anany [23].
Este sería un ejemplo en el que únicamente el *edge* sirve para enviar los fotogramas y el análisis, almacenamiento de fotograma y metadatos se realizará en AWS.
A través de una cámara, en el caso de este proyecto la cámara del portátil, capturar fotogramas de la captura de video, enviar estos fotogramas a través de Kinesis Data Streams a nuestra "pila" montada con CloudFormation que tendrá preparada una función Amazon Lambda para que envié el fotograma a Rekognition y que este reconozca objetos en los fotogramas con la API DetectLabels, asociar una alerta en función de los objetos detectados para que envíe notificaciones a través de SNS y que este envíe un sms al móvil en el caso de detectar uno de ellos.
También podremos visualizar estos fotogramas vía web alojando estos primero en un Bucket S3. Las etiquetas proporcionadas por Rekognition se guardarán en una tabla de DynamoDB y podremos conectarnos a ella, para ver todo esto en un navegador web gracias a Amazon API Gateway y otra función Amazon Lambda.

Casos de uso:

La aplicación apunta a un caso de uso muy específico que es el alertar cuando un objeto, persona o lo que escojamos como parámetro aparece a través de nuestra cámara y este es reconocido por Rekognition. La cámara que puede ser una cámara ip o en el caso del proyecto la cámara del portátil.

1. La cámara del portátil apunta hacia una dirección.
2. El cliente captura el video y envía los fotogramas a AWS. El ratio de fotogramas enviados por segundo se define en la ejecución, ahí serán analizados y almacenados.
3. Si Amazon Rekognition detecta el objeto que hayamos puesto en una lista la función Amazon Lambda ejecuta una función Amazon SNS para que el móvil escogido reciba un sms como alerta.
4. La aplicación dispone también de la posibilidad de ver el análisis de los fotogramas enviados con todas las etiquetas detectadas y la etiqueta seleccionada como alerta a través del navegador web.

6.2.1- Arquitectura de la aplicación:

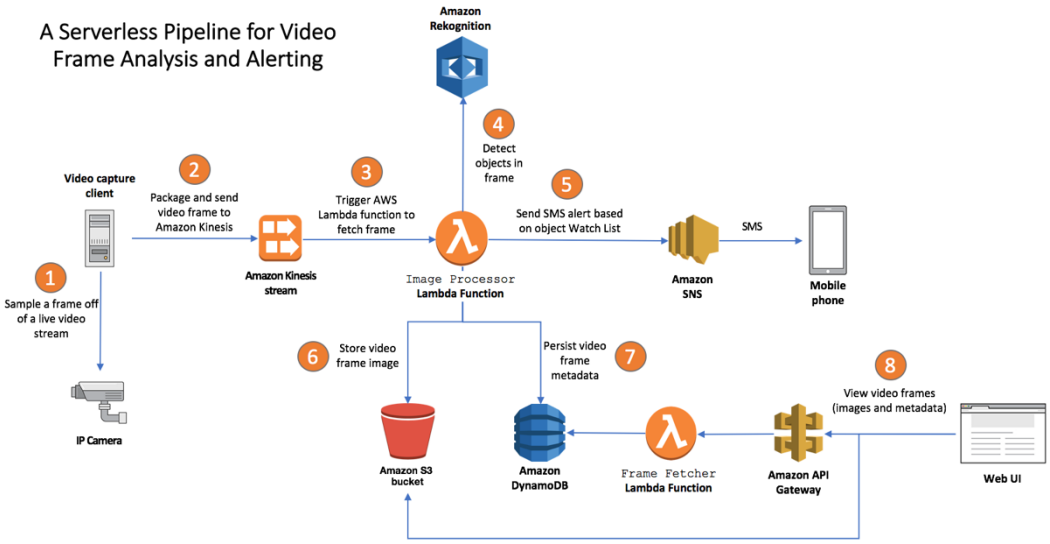


Figura 8: esquema de alerta de análisis video fotograma.

Paso 1 Captura del video y extracción del fotograma:

Inicialmente la cámara del portátil captura los fotogramas a través de OpenCV, el cliente es un script python que se ejecuta en el mismo portátil donde se encuentra la cámara. El script acepta como parámetro el ratio de fotogramas que deseamos capturar. Por ejemplo, si ponemos 20 esto hará que se envíe un fotograma de cada 20. Por defecto el ratio es uno de cada 30 fotogramas.

Paso 2 Empaquetado del fotograma y envío de este a través de Kinesis:

Una vez capturado el fotograma se empaqueta en un objeto en formato JPEG, hay que recordar que Rekognition solo trabaja con JPEG o PNG, también se le añade el momento en que se capturó el fotograma con un *timestamp* asociado al nombre del archivo. El objeto es serializado y enviado a través de Amazon Kinesis Data Stream.

Paso 3 Recibir el fotograma desde Kinesis a la función Lambda:

Una vez empaquetado y enviado el fotograma es recibido por la función Lambda que la hemos llamado “Image Processor” esto es un script escrito en python que se mantiene a la espera de un evento en este caso el recibir el fotograma. Cuando lo recibe se encarga de desempaquetarlo y transformarlo en un objeto python.

Paso 4 La función lambda “Image Processor” detecta objetos en los fotogramas:

La función “Image Processor” envía los fotogramas en formato .jpeg a Amazon Rekognition. Este detecta los objetos del fotograma devolviendo una lista de etiquetas con los objetos encontrados. Cada etiqueta tendrá una puntuación de confianza de como de seguro está Rekognition de la etiqueta en tanto por ciento.

Paso 5 Que ocurre si "Image Processor" detecta que uno de los objetos detectados esta en nuestra lista de alerta:

Dentro de nuestros ficheros de configuración tenemos uno donde ponemos una lista de objetos que si son detectados debe saltar una alerta y recibir un sms si estos tienen un nivel de confianza que también es un parámetro que configurar, por ejemplo, descartar los que no superen el 80% de confianza. La función Lambda "Image Processor" itera sobre las etiquetas devueltas por Rekognition y las compara con nuestra lista de alerta y el nivel de confianza especificado. Si se cumplen estas condiciones "Image Processor" usará Amazon Simple Notification (SNS) para enviar un SMS al móvil que hemos especificado en nuestro json de configuración.

Paso 6 Almacenado de cada fotograma en Amazon S3 a través de "Image Processor":

El siguiente paso es el almacenar cada fotograma capturado en jpeg en un bucked de Amazon S3. La función "Image Processor" se encarga de generar la *key* con que cada fotograma será almacenado. El formato será, año actual / mes actual / día actual / hora actual / UUID generado por python y la extensión .jpg
El nombre del bucked será previamente configurado en el json de configuración, por defecto será "fotogramas".

Paso 7 Los metadatos generados por "Image Processor" de cada fotograma son almacenados en una Base de datos DynamoDB:

Por cada fotograma "Image Processor" almacena los metadatos en Amazon DynamoDB. Estos metadatos incluyen atributos únicos por cada fotograma, en que S3 bucket está almacenado, la *key* donde está la imagen en jpeg almacenada, el tiempo aproximado de cuando fue capturado, la lista de etiquetas y su porcentaje de confidencia que ha dado Amazon Rekognition.

Paso 8 visualización de los fotogramas y sus metadatos a través del navegador web:

El último paso es visualizar a través de una interfaz simple basada en web los fotogramas y sus metadatos.
La web periódicamente invoca otra función Amazon Lambda llamada "*Frame Fetcher*". Esta función hace *queries* a DynamoDB y devuelve los fotogramas más recientes junto sus metadatos. También se encarga de generar URL para cada fotograma almacenado en el bucked S3 para que la Web pueda desplegar las imágenes y así poder visualizarlas.

Explicación de la implementación.
Para poder implementar el proyecto se necesita:

1. Registrarte en https://aws.amazon.com/es/ y crear un usuario Administrador.
2. Tener instalada la versión 2.7 de python y pip.
3. Crear con virtualenv un entorno virtual de Python y así poder aislar el entorno.
4. Usar pip para instalar AWS CLI y configurarlo. AWS Command Line Interface [20] es una herramienta open source que te permite interactuar con AWS usando la línea de comandos en la propia Shell del portátil. Con una mínima configuración podemos

acceder a todas las funciones de aws sin necesidad de acceder a la consola de AWS vía web.

5. Establecer una región predeterminada todas los servicios deben estar alojados en la misma región de lo contrario no funcionará, en el caso del proyecto se decide que será Irlanda (eu-west-1) por proximidad y porque tiene Amazon Kinesis Video Stream[40]
6. Usar pip para instalar Open CV 3 se usará para la captura de fotogramas del video.
7. Usar pip para instalar Boto3, esto es un SDK para python de AWS que permite a los desarrolladores escribir software para los servicios de Amazon.
8. Usar pip para instalar Pynt.
9. Clonar el repositorio.
10. Usar pip para instalar pytz. Pytz es necesario para calcular las zonas horarias.

6.2.2 Proceso de despliegue del proyecto en AWS.

Flujo de despliegue con los diferentes comandos a ejecutar en el orden indicado:

1. Comando packagelambda, empaquetamos las funciones “Image Processor” y “Frame Fetcher” en .zip separados junto con sus respectivos json de configuración.
2. Comando deploylambda, subirá a dos bucked S3 los correspondientes .zip del “Image Processor” y “Fotograma Fetcher”.
3. Comando createstack, creará a través del template aws-infra-cfn.yaml una pila de CloudFormation que desplegará todo lo necesario para poder ejecutar todo el flujo del proyecto.
4. Comando webui monta la UI Web
5. En otro terminal ejecutamos el comando webuiserver para montar el servidor http basado en python.
6. Por último el comando “videocapture[20]” con esto iniciaremos la captura de los fotogramas y el flujo a través de toda la arquitectura, el numero 20 es un ejemplo en este caso se quedaría con uno de cada 20 fotogramas del video capturado por la cámara del portátil.

6.2.3 Flujo de ejecución de la implementación

1- Captura del fotograma y envío de este a través de Kinesis Data Stream:

Aquí capturamos los fotogramas gracias a Open CV usamos pytz para obtener la hora y la zona horaria del momento de la captura y así poder empaquetarla junto al fotograma.
Con los clientes de Kinesis iniciados procedemos a enviar los datos con la función: kinesis_client.put_record().

En este caso y en la misma función de captura de fotogramas “videocapture[20]” podríamos enviar la imagen directamente a Rekognition y recibir los resultados de su análisis primero iniciando el objeto cliente de Rekognition y después con la función detect_label():

```
rekog_client.detect_labels(
  Image={
    'Bytes': img_bytes
  },
```

```
    MaxLabels=rekog_max_labels,
    MinConfidence=rekog_min_conf
)
```

2- Procesado de la imagen en la función Lambda Image Processor:

En esta función Amazon Lambda que está configurada a través de CloudFormation para que cada vez que se produzca un evento, en este caso el enviar un fotograma con Kinesis Data Stream con un nombre en concreto, accedemos a las etiquetas que nos devuelve Rekognition a través de la función detect_labels().
Las partes más importantes de los json que devuelve serían Nombre de la etiqueta (puede ser una persona, cara, objeto, texto, o un escenario como por ejemplo tráfico), nivel de confianza en porcentaje y el BoundingBox de donde se encuentra.

Hacemos la comprobación para ver si tiene alguna coincidencia con nuestra lista de alerta. De estar dentro una de las etiquetas iniciaríamos el proceso de enviar el sms al número indicado en el fichero de configuración con la función:
sns_client.publish(PhoneNumber=label_watch_phone_num, Message=notification_txt)

Independientemente de si hay una etiqueta que coincide con nuestra lista de objetos de alerta como si no el fotograma se almacena en el S3 bucked:

```
s3_client.put_object(
    Bucket=s3_bucket_name,
    Key=s3_key,
    Body=img_bytes
)
```

También sus metadatos en DynamoDB:

```
item = {
    'fotograma_id': fotograma_id,
    'processed_timestamp' : processed_timestamp,
    'approx_capture_timestamp' : approx_capture_timestamp,
    'rekog_labels' : rekog_response['Labels'],
    'rekog_orientation_correction' :
        rekog_response['OrientationCorrection']
        if 'OrientationCorrection' in rekog_response else 'ROTATE_0',
    'processed_year_month' : year + mon,    's3_bucket' : s3_bucket,
    's3_key' : s3_key
}

ddb_table.put_item(Item=item)
```

3- Visualización del resultado del análisis en el navegador web:

El servidor web que tenemos montado nos permite visualizar las imágenes con las etiquetas que estas generan.
Para ello usamos Amazon API Gateway que trabaja entre los dos nuestro servidor web y AWS, este hace peticiones GET a una dirección en concreto y la función Lambda “Frame Fetcher” está preparada para responder al evento de peticiones GET, una vez recibida se encargará de

hacer las *querys* a DYnamoDB y de buscar la imagen correspondiente para enviarla devuelta a API Gateway y de esta a nuestra web.

6.2- Detección de caras en una señal streaming de video:

La implementación es una adaptación del ejemplo proporcionado por Amazon y otro que ha hecho el usuario Young Yang de GitHub [44].

El análisis de caras no entra dentro de nuestro caso de uso por el que estamos buscando solución en este trabajo, pero es lo único que permite hacer Amazon Rekognition por medio de un video en directo con Amazon Kinesis Video Streams. CreateStreamProcessor solo permite hacer detección de caras con una colección. Pero puede que en un futuro próximo AWS incluya más funcionalidades a la API CreateStreamProcessor y por eso creo conveniente trabajarla en este apartado. Así conocer su funcionamiento, ejecución y latencias.
El *BoundingBox* que devuelve cuando detecta una cara que se encuentra en nuestra colección es el de la imagen de la colección y no la del fotograma del video.
Esta solución nos proporcionará el nombre de la persona que aparece en el *stream* si su cara está en la colección y si Rekognition la reconoce. Se intenta comparar que opción es mejor, por una parte, se analiza cada fotograma de forma local en el *edge* a través de OpenCV y las librerías de detección de caras y ojos "Haar cascade" [45] y se coloca un rectángulo en esta y en los ojos. Por otra la proporcionada por Amazon Rekognition que en este caso además de dar la posición donde se encuentra la cara y los ojos en el fotograma también nos dice si la cara que aparece en el fotograma está en la colección y así detectar una persona especifica, aunque repito no así su posición en el video.

Los pasos necesarios son los siguientes:

1. Crear un servicio Kinesis Video Stream.
2. Crear un servicio Kinesis Data Stream.
3. Crear la colección donde vamos a depositar las caras que queremos que detecte.
4. Subir las imágenes que contengan las caras en la colección.
5. Crear el Stream Processor asociándole como input el Kinesis Video Stream, también asociar como output el Kinesis Data Stream, poner en el apartado settings la acción FaceSearch junto con la colección. Y finalmente el rol con los permisos para poder ejecutar los servicios.

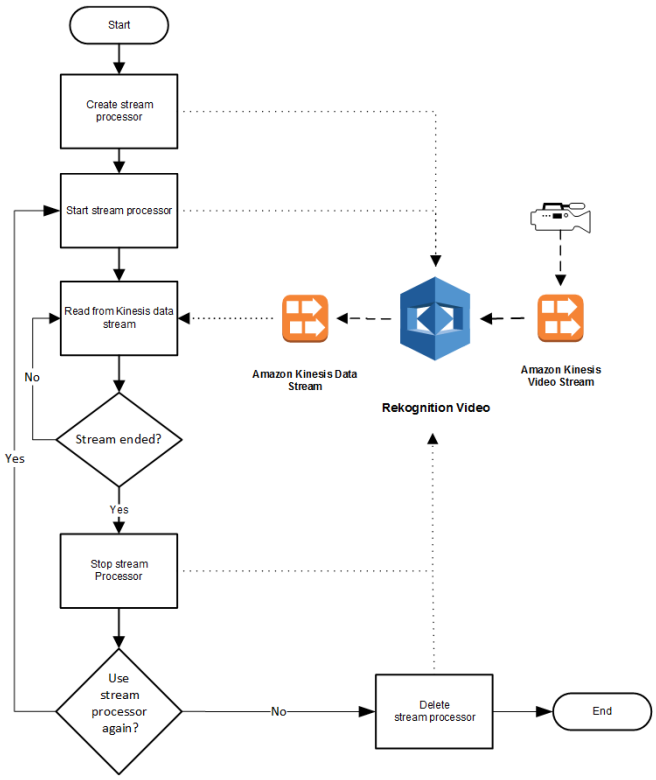


Figura 9: Esquema de análisis de caras con un Stream Processor.

Se puede hacer detección de caras de un video sin necesidad de subir el video entero, si no fotograma por fotograma esto simplifica muchísimo las cosas pues no es necesario crear una función Kinesis Video Stream, Kinesis Data Stream ni un Stream processor. Únicamente enviando el fotograma directamente a Rekognition con la API DetectFaces esta nos devolverá el json con toda la información. Esto serviría también para el ejemplo anterior donde enviábamos el fotograma a través de KDS para que la función Lambda hiciera el resto del trabajo. Pero se hubiera podido reducir a enviar el fotograma directamente a Rekognition con la API LabelDetection y una vez obtienes las etiquetas y de encontrarse un objeto de la lista de alertas hacer la llamada a Amazon SNS y que este enviara el sms al móvil.
En el caso de detectar caras específicas de una colección no sería posible de esta manera.

6.3- Uso de las diferentes APIs de rekognition en videos alojados en S3

Como hemos explicado anteriormente en el análisis, el hecho de analizar un video que ya está almacenado es contrario a la idea de hacer análisis de video en directo. Pero puede haber casos de uso en los que puede ser útil además de aprender el uso de SNS junto con SQS.
En esta prueba usaremos Amazon Rekognition Video para procesar vídeo que tendremos almacenados en un bucket de Amazon S3.
Usaremos dos videos bien diferenciados para realizar las pruebas:

1. Tráiler de la película Casino Royale [24].
2. Partido de baloncesto de 10 min de duración [25].

Se probarán diferentes APIs de rekognition:

- StartLabelDetection: usaremos la API de detección de etiquetas para que nos muestre en que fotogramas aparece un arma de fuego.
- StartPersonTracking: mostraremos las personas que aparecen en el video del partido de baloncesto y a través de los índices un seguimiento de las personas.
- StartCelebrityRecognition: en el tráiler de la película de 007 averiguaremos todos los famosos que aparecen y en qué momento lo hacen.

El escenario de diseño es un conjunto asíncrono de operaciones. Esto quiere decir que el análisis del video comienza cuando se llama a una operación como por ejemplo StartCelebrityRecognition esta devuelve un JobId. El estado de la realización del análisis se publica en un tema de Amazon SNS, para obtener el estado de realización del tema de SNS usaremos una cola SQS. Cuando finalice el análisis empezaremos con operación Get para obtener las etiquetas, que en el caso de StartCelebrityRecognition sería GetCelebrityRecognition.
De una forma manual se podría pensar en realizar la petición Get para obtener los resultados con el JobId que devuelve Rekognition, con por ejemplo un bucle que iterara de forma infinita hasta que el trabajo estuviera hecho y así no tener que montar la estructura de SNS y SQS pero esto no es posible porque Amazon Rekognition limita el número de veces que se puede hacer una petición Get de Rekognition.

En el siguiente diagrama se muestra el flujo para detectar etiquetas (StartLabelDetection) en un video almacenado en S3:

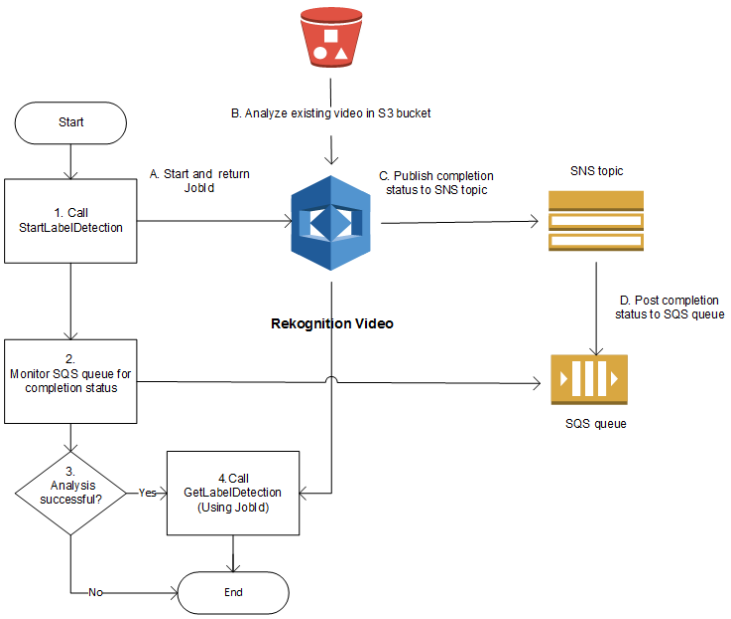


Figura 10: Esquema de estructura de análisis de videos almacenados en S3

Para implementar este modelo se ha usado python con las librerías Boto3.
En el script startStop.py se usará para inicializar todos los servicios necesarios para poder realizar la tarea del análisis de video almacenado. El usuario debe tener los siguientes permisos:

1- Leer y poder crear buckets S3
2- Leer/Escribir/Crear temas SNS

3- Leer/Escribir/Crear colas SQS
4- Acceso a Rekognition
5- Rekognition tiene que poder escribir en un tema SNS

Los servicios que se inician son los siguientes:

1. Crear el "tema" de SNS
2. Crear la cola SQS
3. Crear el Bucket de S3 donde alojaremos los videos
4. Subscribir el tema SNS a la cola SQS, en este paso es importante dotar de permisos a la cola SQS para que se pueda escribir en ella.

El script de python de ejecución es videoProcessor.py se realizarán las siguientes acciones:

1- Comenzar la llamada al inicio de detección de etiquetas, de famosos o de personas. En esta llamada se especifica en los parámetros el bucket donde se encuentra el video y el video en particular, el tema SNS que servirá para notificar cuando Rekognition ha terminado de analizar el video y el rol con los permisos para poder enviar permisos de Rekognition a SNS.
2- Al iniciar el análisis y llamar a la función Start nos devolverá una respuesta de la que entre todos los parámetros nos quedaremos con el JobID que nos servirá para más adelante.
3- Sondeamos la cola SQS que está suscrita a el tema SNS anterior para recuperar el mensaje conforme se ha terminado el análisis del video. Para ello permanecemos en un bucle hasta que la respuesta de SQS contiene el campo Messages. Una vez recibido se comprueba que pertenece al mismo JobID guardado anteriormente como resultado del inicio del análisis de Rekognition.
4- Cuando el JobID que encontramos en SQS coincide con el devuelto por Rekognition pasamos a la función donde recogeremos los datos devueltos del análisis gracias a las funciones Get. Las funciones Get tienen como parámetros de entrada:
    a. JobId.
    b. MaxResults, donde indicaremos el máximo de resultados por json que queremos recibir.
    c. NextToken, en caso de que haya más resultados de los que hemos permitido recibir podremos acceder a la siguiente página con este parámetro.

```
response = self.rek.get_celebrity_recognition(JobId=jobId,
                                              MaxResults=maxResults,
                                              NextToken=paginationToken)
```

El json con la información del análisis del video donde encontraremos la información del video (el nombre del famoso, las coordenadas donde se encuentra esa persona…), un parámetro NextToken por si se han quedado resultados por mostrar en la página, que pondremos en el parámetro de la función Get.

5- En esta función imprimiremos por pantalla los resultados que más nos convengan. Como por ejemplo que nos muestre en que fotogramas aparece un arma en el tráiler de "*Casino Royale"*.

Otro método probado como análisis de video sería sin utilizar un video almacenado si no un video en el *edge*. A través de OpenCV coger fotograma a fotograma del video y usar el Rekognition de imágenes y utilizar el resultado a conveniencia.

En este caso no es necesario:

1. Crear el “tema” de SNS
2. Crear la cola SQS
3. Crear el Bucket S3 donde alojaremos los videos
4. Subscribir el tema SNS a la cola SQS, en este paso es importante dotar de permisos a la cola SQS para que se pueda escribir en ella.

Solo necesitamos tener permisos para acceder a Rekognition, aunque en este caso las funciones de Person Traking no serían posibles pues analiza imágenes de forma individual y no dentro de un contexto.

Para implementar un ejemplo se ha usado el de búsqueda de famosos.

- Se crea un video de salida en el que guardaremos los resultados
- Se captura fotograma a fotograma el video del tráiler de Casino Royale.
- Se envía cada fotograma de forma individual a Rekognition. Con la función API RecognizeCelebrities.
- En el caso de encontrar un famoso se imprimirá en el video de salida el *BoudingBox* donde se encuentra el famoso, el nombre del famoso y el porcentaje de confianza que tiene Rekognition sobre esa decisión.

6.4- Identificación de objetos en transmisiones de vídeo con Amazon SageMaker

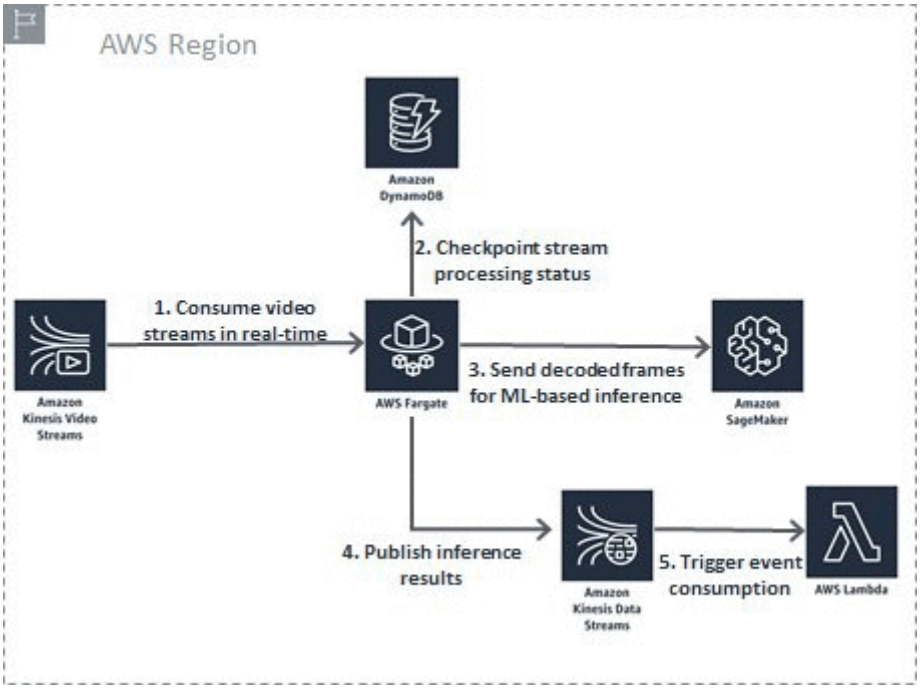


Figura 11: Esquema Identificación objetos con SageMaker

Ejemplo de implementación de detección de objetos en un *streaming* de video con Amazon Kinesis Video Stream a través de un modelo entrenado con Amazon SageMaker creado por Aditya Krishnan y Jagadeesh Pusapadi [26].

Con la ayuda de un contenedor Docker que en la que se incluye la funcionalidad de la aplicación y una plantilla de AWS CloudFormation. El template de AWS CloudFormation automatiza el despliegue de toda la infraestructura necesaria de AWS en nuestra cuenta,

desde la lectura del video mediante Kinesis Video Streams hasta la invocación del *endpoint* de Amazon SageMaker para el análisis basado en Machine Learning.

6.4.1- Recursos de AWS que la plantilla de CloudFormation crea:

- Un clúster Amazon Elastic Container Service (ECS) que utiliza el motor informatico AWS Fargate que ejecuta el software de la biblioteca. Esto permite ejecutar el software alojado en un container Docker.
- Una tabla Amazon DynamoDB que mantiene los puntos de comprobación (checkpoints) y el estado. Todo lo relacionado con los trabajos que usan Fargate Tasks y Amazon Kinesis Data Streams para capturar los resultados que genera Amazon SageMaker.
- Un Kinesis Data Stream para capturar los resultados de inferencia que se genera desde Amazon SageMaker.
- Una función de AWS Lambda que analiza la salida de Amazon SageMaker.
- Crear las políticas que se necesitan para acceder a los servicios con AWS Identity y Access Management (IAM).
- Para monitorizar la aplicación creará un recurso de Amazon CloudWatch.

6.4.2- Antes de desplegar la plantilla de Amazon CloudFormation se requieren 3 pasos.

1. Tener instalado un productor con el que enviar el video a través de Amazon Kinesis Video Stream. En este caso seguiremos usando la solución proporcionada por Amazon que usa GStreamer.
2. Que la cuenta que usamos tenga vinculado un rol de acceso a los servicios para que AWS Fargate funcione.
3. Crear un Bucket S3 donde se almacenará nuestro modelo.
4. Tener preparado el modelo ya entrenado y listo con el *endpoint* definido. Para este ejemplo se ha usado el algoritmo de ejemplo proporcionado por Amazon de detección de objetos, pero se podría usar cualquier otro.

6.4.3- Detección de objetos con SageMaker:

Para poder ejecutar la plantilla de jupyter notebook, hacer el entrenamiento y finalmente desplegar el *endpoint* serán necesarias tres instancias de Amazon SageMaker.

Las utilizadas para esta implementación han sido:

Tipo de instancia	Uso	CPU Virtual	GPU	Memoria (GiB)	Memoría GPU (Gib)	Rendimiento de red
ml.t2.medium	Ejecutar notebook	2	0	4	-	De bajo a moderado
ml.p3.2xlarge	Entrenamiento	8	1xV100	61	16	Hasta 10 Gbps

ml.m4.xlarge	endpoint	4	-	16	-	Alto

Nota: Para las instancias ml.p3.2xlarge y ml.m4.xlarge será necesario contactar con el soporte técnico para que aumente de 0 a 1 la posibilidad de desplegar esas instancias. En los tipos de instancias SageMaker Training y SageMaker Hosts respectivamente.

6.4.4- Explicación del modelo usado**:**

Para empezar, tendremos que crear una instancia de SageMaker de tipo notebook donde ejecutaremos con el jupyter notebook object_detection_image_json_format.ipynb

Dentro encontraremos todos los pasos necesarios para poder crear nuestro modelo.

El notebook es un ejemplo *end-to-end* que sirve para mostrar las capacidades de SageMaker en los algoritmos de detección de objetos. En esta demo se demostrará como entrenar usando el modelo en el dataset COCO [42] usando el algoritmo Single Shot multibox Detector (SSD) [43]. También sirve para ver cómo construir un dataset de entrenamiento usando el formato JSON.

MS COCO es un *dataset* de gran escala que sirve para múltiples tareas de visión por computación, como detección de objetos, segmentación.

Los primeros pasos son los de definir las variables como el rol del usuario y el bucket S3 donde se guardará todo y que hemos creado previamente. Después la preparación de los datos y darles formato.
El propio notebook proporciona las funciones para limpiar y dar formato JSON al dataset.
Para esta tarea se usará el data set MS COCO de 2017 que contiene 5000 imágenes con objetos de 80 categorías.
El siguiente paso es subir los datos a al bucket S3.

6.4.5- Entrenamiento**:**

Para empezar con el entramiento crearemos el objeto sageMaker.estimator.Estimator con los siguientes parámetros de entrada [27]:

- role: AWS IAM role con acceso a SageMaker y al endpoint que después usaremos
- train_instance_count=1: el número de instancias EC2 que usaremos para entrenar en este caso 1.
- train_instance_type='ml.p3.2xlarge': el tipo de instancia que usaremos en este caso la más económica con GPU.
- train_volume_size = 50: tamaño en GB de volumen EBS para almacenar los datos de input durante el entrenamiento. En este caso se le asignan 50GB.
- train_max_run = 360000: el timeout en segundos por entrenamiento. Si se supera el tiempo SageMaker terminará con el entrenamiento independientemente del estado en el que esté.
- input_mode = 'File': el “input mode” que el algoritmo soporta. En este caso “File” que quiere decir que Amazon SageMaker copia el dataset de entrenamiento desde buket S3 a un directorio local.
- output_path=s3_output_location: la localización del buket S3 donde se guarda el resultado del entrenamiento.
- sagemaker_session=sess: objeto de sesión SageMaker donde se gestiona las interacciones con las APIs y servicios de AWS necesarios.

El algoritmo de detección de objetos usado es el Single-Shot Multi-Box detection (SSD). Donde tendremos que configurar varios hyperparametros que nos ayudarán a estimar parámetros del modelo desde el *dataset* de entrenamiento. Estos hyperparametros son [28]:

- num_classes=80: con este parámetro definimos la dimensión de las clases de salida que es la misma que la del *dataset*, 80.
- num_training_samples=4452: el número de ejemplos de entrenamiento del *dataset* de entrada. 4452 ejemplos.
- base_network='resnet-50': la arquitectura de red base a usar.
- use_pretrained_model=1: poniendo a 1 se cargará el modelo pre entrenado para el entrenamiento.
- image_shape=512: el tamaño de las imágenes de entrada. Se reescalarán las imágenes al tamaño de 512.
- label_width=600
- learning_rate=0.001: el ratio inicial de aprendizaje.
- mini_batch_size=16: el tamaño del lote de entrenamiento. En una configuración de una maquina multi-gpu, cada GPU maneja mini_batch_size/num_gpu muestras de entrenamiento.
- optimizar='sgd': los diferentes tipos de optimización de la API MXNET
- momentum=0.9: el momentum para 'sgd'.
- nms_threshold=0.45: el threshold de supresión no máximo.
- overlap_threshold=0.5: el threshold de coincidencia de la evaluación
- weight_decay=0.0005: el coeficiente de disminución de peso para 'sgd'
- epoch=30: esto define cuantos pasos del *dataset* se itera y determinar el tiempo del algoritmo.

Al terminar este paso hay que preparar la conexión entre el algoritmo y los datos que se encuentran en los canales dentro del buket S3. Y definir el formato de las imágenes que usaremos en este caso JPEG.
Como último paso del entrenamiento ejecutaremos:

```
od_model.fit(inputs=data_channels, logs=True)
```

El resultado del entrenamiento se almacenará en el buket S3.

6.4.6- Hosting (creación endpoint)

Cuando el entrenamiento concluya, podremos desplegar el modelo entrenado como un *endpoint* en tiempo real de SageMaker. Con este *endpoint* podremos predecir desde el modelo. No se puede escoger la misma instancia ni del mismo tipo para entrenar como para el host. Se puede usar de instancia para el host una más barata sin GPU. En este caso se usará ml.m4.xlarge.

Una vez tengamos el *endpoint* pasamos a crear la pila con Amazon CloudFormation.

Seleccionamos la plantilla en la región que mejor nos convenga en este caso usaremos el de Irlanda:

- Inicio en la región UE (Irlanda) (eu-west-1)

La plantilla viene con los siguientes campos:

1. Nombre de la pila: aquí pondremos el nombre de la pila.
2. AppName: este debe ser único por defecto aparece “KVS-SageMaker-Driver”
3. DockerImageRepository: la imagen Docker donde está alojado todo lo necesario para que funcione Kinesis Video Streams junto con la integración del Driver de SageMaker.
4. EndPointAcceptContentType: el tipo de contenido que acepta nuestro *endpoint* y que hemos definido en la etapa de la instancia notebook. En este caso image/jpeg.
5. LambdaFunctionBucket: el nombre del bucket S3 donde se alojará la función lambda si deseamos introducir una.
6. LambdaFuntionKey: el nombre.zip con se que guardará esa función lambda.
7. SageMakerEndpoint: el nombre del *endpoint* que hemos creado previamente que va a SageMaker. NO el ARN del *endpoint*.
8. StreamNames: aquí especificaremos los nombres de los streams de Kinesis Video Strems, en este caso solo usaremos uno. De poner más los separaríamos por comas.
9. TagFilters: las etiquetas de los filtros que nos da el JSON aquí se deja el texto que viene predefinido.
10. Seleccionar la casilla conforme CloudFormation va a crear recursos de IAM, al final de la página.

Dentro del contenedor Docker que crea con la plantilla de CloudFormation se pueden modificar varios parámetros, el más importante para este caso de uso sería el campo inferenceInterval que tiene predeterminado el valor “6” es te valor significa que se coge 1 fotograma por cada 6 y se envían al *endpoint* de SageMaker. Para modificar estos parámetros hay que hacerlo antes de desplegar la pila y modificando la plantilla.

6.4.7- Monitorización y visualización mediante Dashboard con CloudWatch

Una vez desplegada la pila podremos monitorizar las métricas de la aplicación para ello usaremos los Dashboards de Amazon CloudWatch, la información a mostrar ya está configurada por la plantilla y sería la siguiente:

- Fotograma Metrics: métricas para procesar la transmisión de vídeo, enviar fotogramas al *endpoint* de Amazon SageMaker y la escritura de la salida de información de esta a la función Lambda por medio de Kinesis Data Streams.
- IngestToProcessLatency: la diferencia de tiempo entre que se introduce un fotograma en Kinesis Video Streams y el momento que la aplicación lo recibe.
- Current Lease Total: el total de asignaciones actuales de Kinesis Video Streams activas. En nuestro de caso de uso será uno.
- Lease Sync Metrics: metricas de sincronización de transmisiones.
- LeaseCount por Worker: recuento de asignaciones por proceso de trabajo.
- Number of Workers: número de procesos de trabajo.

- ECS Service Utilization: métricas de uso del clúster de Amazon ECS.
- KinesisDataStream: métricas de este servicio.
- SageMaker: operaciones realizadas por el bloc de notas de Amazon SageMaker.
- Lambda: número y duración de la función Lambda que procesa la salida del bloc de notas de Amazon SageMaker.

Para ver la detección de objetos que realiza SageMaker usaremos los registros de la función Lambda. Los pasos para acceder a ella son los siguientes:

1. Abrir la consola AWS Lambda
2. Escoger la función Lambda de nuestro programa.
3. Seleccionar “Monitorización”.
4. Finalmente, en “Ver registros en CloudWatch”.

6.5- Implementación de DeepStream En una instancia de Amazon EC2

El uso de DeepStream no se ha probado en este proyecto final de grado. Lo que se explica a continuación sería como desplegar DeepStream en una instancia EC2 y como poder trabajar con ella. Para implementarlo primero tendremos que lanzar una instancia en AWS EC2 que tenga GPU de Nvidia.

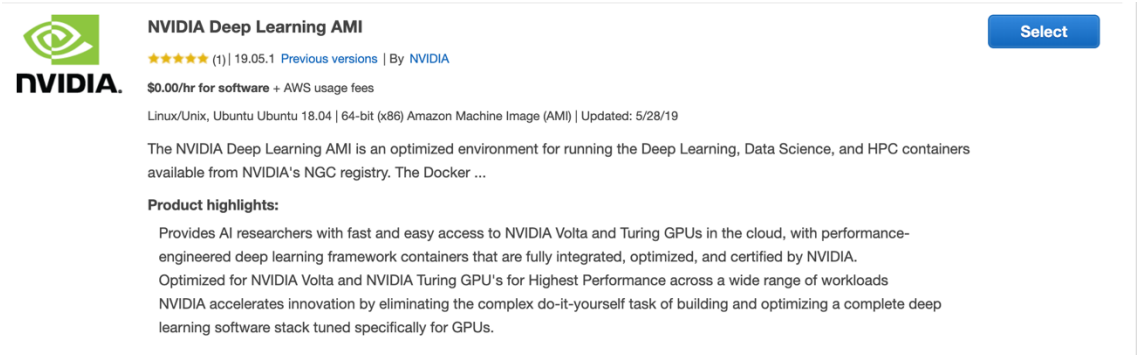


Figura 12: NVIDIA Deep Learning AMI

La instancia NVIDIA Deep Learning AMI viene con todo lo necesario para poder usar DeepStream.

Cargar el contenedor docker en la instancia con DeepStream y todo lo necesario:
`docker pull nvcr.io/nvidia/deepstream:3.0-18.11`

Una vez instalado para poder conectarnos vía SSH hay que hacer lo siguiente [29]:

- En una ventana de terminal, utilice el comando ssh para conectarse a la instancia. Especificar el archivo (.pem) de clave privada

y *nombre_de_usuario@nombre_dns_público*. la AMI de NVIDIA Deep Learning, el nombre de usuario es ec2-user.

ssh -i /path/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com

Para poder acceder a los videos que deseamos analizar y que tengamos alojados en S3 lo podemos hacer con AWS CLI que ya viene instalada por defecto en la instancia:

Para copiar un archivo de S3 a la instancia:

- aws s3 cp s3://my_bucket/my_folder/my_file.ext my_copied_file.ext

Para copiar el resultado o cualquier objeto de una instancia a S3:

- aws s3 cp my_copied_file.ext s3://my_bucket/my_folder/my_file.ext

Con el comando sync podemos sincronizar un bucket de S3 completo en una ubicación de nuestra instancia. Con esto podemos tener la copia de la instancia actualizada con S3.

- aws s3 sync s3://remote_S3_bucket local_directory

Si por el contrario queremos analizar video en *streaming* producido por Amazon Kinesis Video Streams tendremos que usar la API **GetMedia** [30].
Con esta "request", podemos identificar el Amazon Resource Name (ARN) del stream y empezar por el trozo (chunk) que queramos.

Primero debemos llamar a la API GetDataEndPoint para obtener el *endpoint*. Después enviar la "request" GetMedia a este *endpoint* usando el –endponit-url paramater.
Cuando ponemos el contenido del video (fragmentos) en un stream, Kinesis Video Streams almacena cada fragmento que entra con sus metadatos relacionados, esto sería lo que se llama un "chunk" o pedazo. La API de Getmedia devuelve un stream de estos "chunks" empezando por el "chunk" que especifiquemos en la "request".

Los límites de la API GetMedia son:

- Un cliente puede hacer una llamada a GetMedia hasta 5 veces por segundo y por stream.
- Kinesis Video Streams durante una sesión de GetMedia tiene un ancho de banda hasta 25 megabytes por segundo (o 200 megabits por segundo).

En el caso de ocurrir un error después de invocar la API de Kinesis Video Stream, además del código de estado HTTP y la respuesta del "body" se incluye la siguiente información el tipo de error y el id de este.

La sintaxis de la request es:

```
POST /getMedia HTTP/1.1
Content-type: application/json

{
  "StartSelector": {
    "AfterFragmentNumber": "string",
    "ContinuationToken": "string",
    "StartSelectorType": "string",
    "StartTimestamp": number
```

```
  },
  "StreamARN": "string",
  "StreamName": "string"
}
```

7- Pruebas, evaluación, resultados

7.1- Resultado del Face Detection con Amazon Kinesis Video Stream

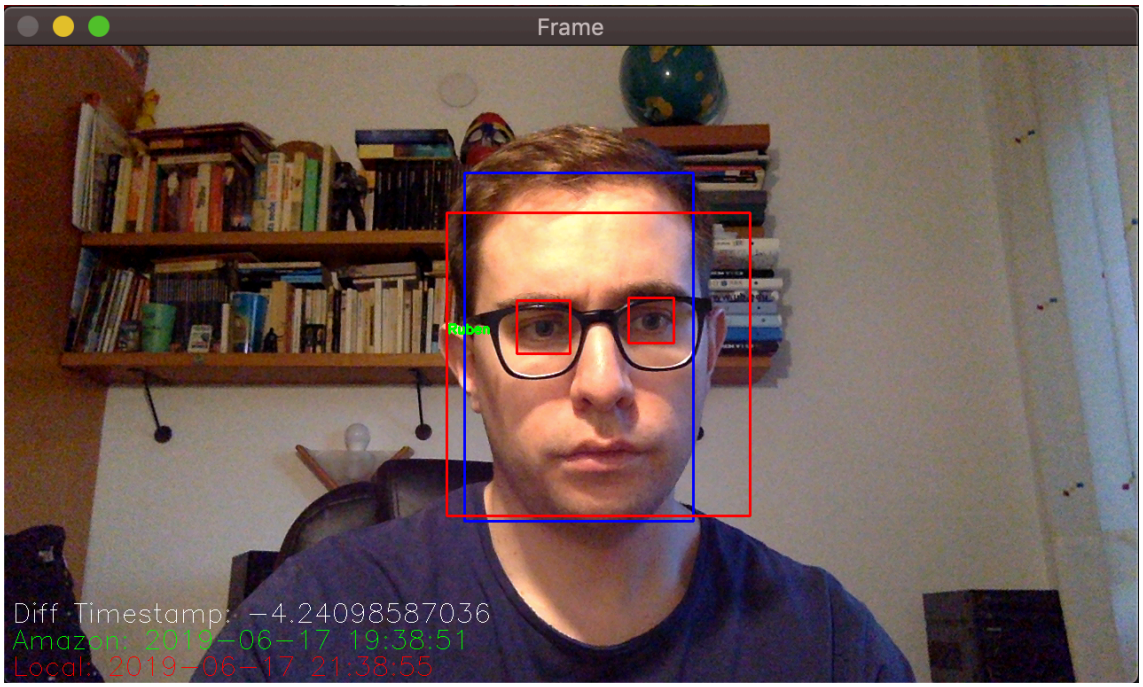


Figura 13: Resultado Face Detection

Ejecutado desde los servidores alojados en la región de Irlanda.
El *BoundingBox* en color azul es el proporcionado por Amazon Rekognition al detectar una cara.
En rojo es el *BoundingBox* de cara y ojos detectado en local por OpenCV con las librerías "Haar Cascade classifier"
Los *timestamps* de la izquierda inferior corresponden:

- Rojo: hora local en la que se captura el fotograma.
- Verde: hora en la que ya se ha procesado por Rekognition
- Blanco: la diferencia entre estos dos

Las diferencias entre los dos rondan siempre entre 4 y 6 segundos usando conexión Wi-Fi.

7.2- Resultado Serverless Solution for Video Fotograma Analysis and Alerting

Al detectar un arma en la imagen la función lambda guardara esa etiqueta como alerta.

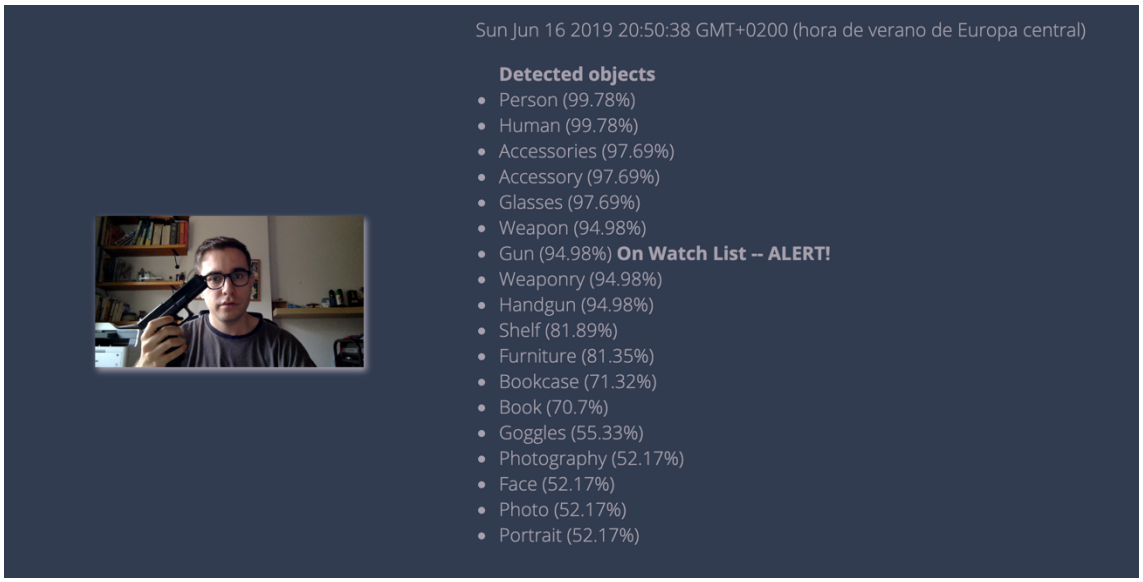


Figura 14: resultado cuando encuentra un arma

Además de activar Amazon SNS para que envíe un sms al móvil configurado

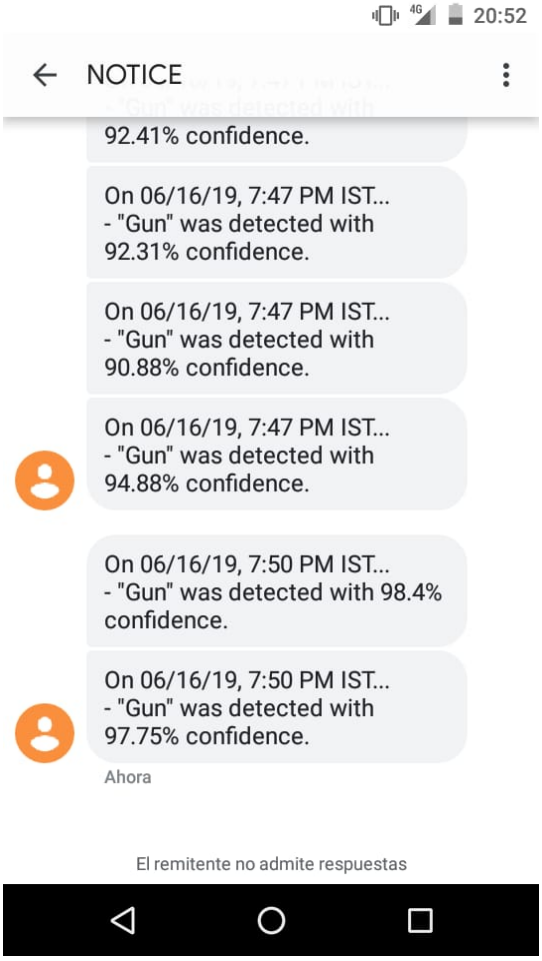


Figura 15: resultado de sms cuando detecta una etiqueta en la lista de alertas

7.3- Resultado de hacer el análisis de Famosos:

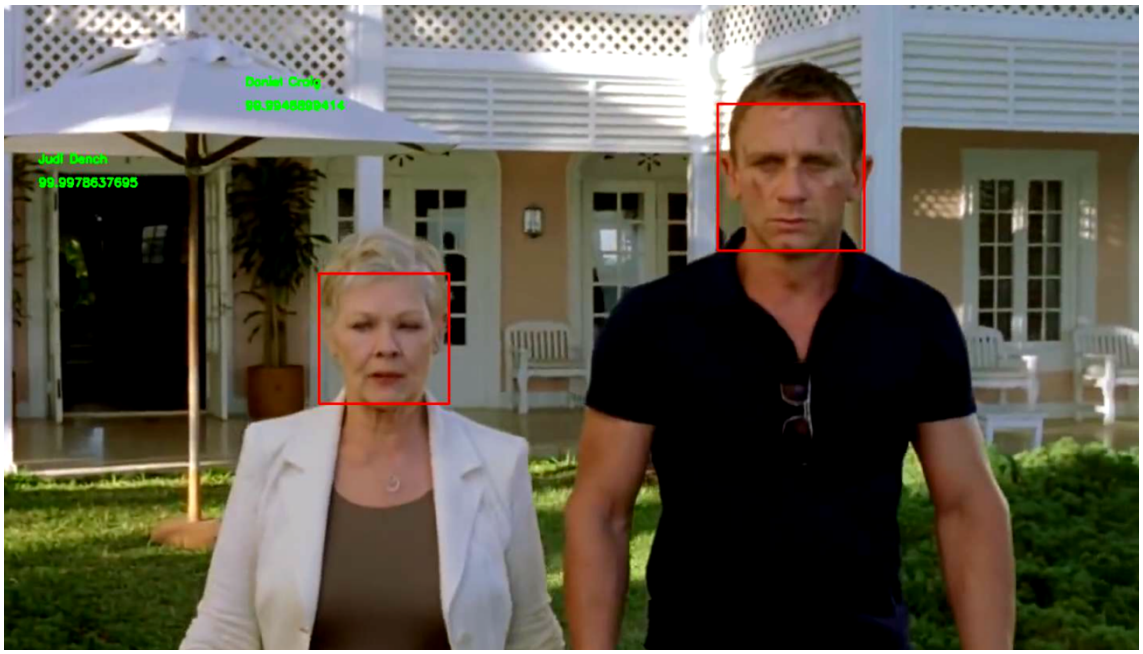


Figura 16: resultado Rekognition Celebrities

El resultado de la figura 16 corresponde a el uso de Amazon Rekognition Imágenes con la API RecognizeCelebrities. En el caso de videos almacenados la función GetCelebrityRecognition devuelve un json que se visualiza por el terminal con toda la información. El *timestamp* en el que aparece el famoso, el nombre del famoso, la posición de su cara…

7.4- Identificación de objetos en transmisiones de vídeo con Amazon SageMaker

El modelo no da los mismos buenos resultados que Amazon Rekognition como era de esperar pues la muestra es inferior a la que han usado Amazon.

Algunos ejemplos que si han funcionado:

Figura 17: mostrando botella a la cámara

Al mostrar una botella de agua devuelve:
detected object: bottle, with confidence: 0.192102149129

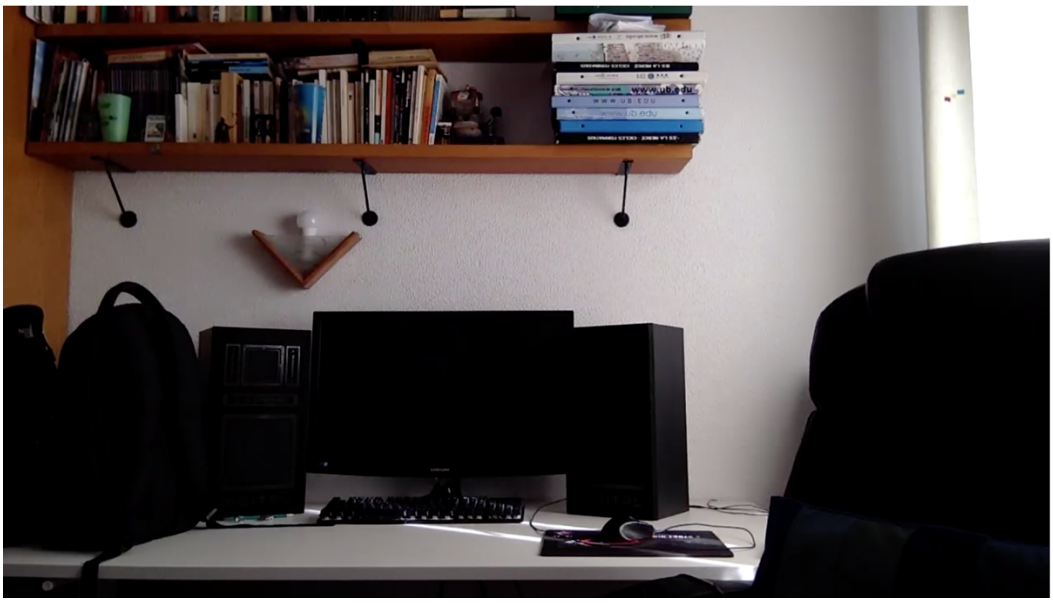


Figura 18: Mostrando el monitor y la silla del escritorio.

El monitor del pc devuelve:
detected object: tvmonitor, with confidence: 0.165770053864

La silla del escritorio:
detected object: chair, with confidence: 0.168395996094

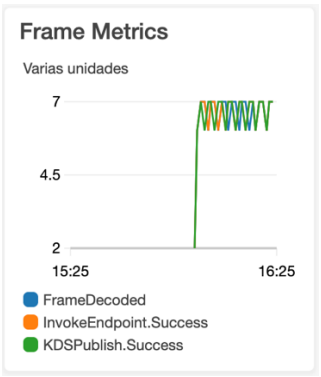
Figura 19: mostrando persona.

Las personas es lo que mejor detecta devolviendo siempre sin problemas:
detected object: person, with confidence: 0.520014286041

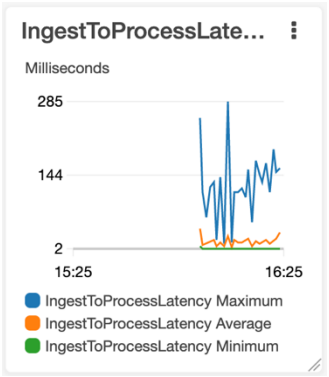
Figura 20: mostrando revista con un coche

Al intentar mostrar tanto coches de juguete como en una revista no detecta nada, pero en el caso de la revista la detecta como book:
detected object: book, with confidence: 0.160073935986

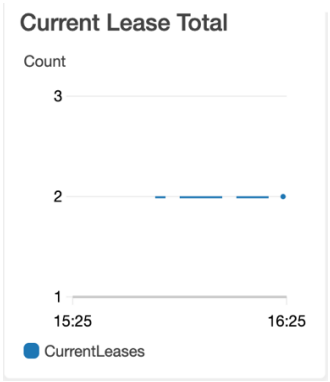
Resultados de la monitorización y visualización en Dashboard de Amazon CloudWatch:



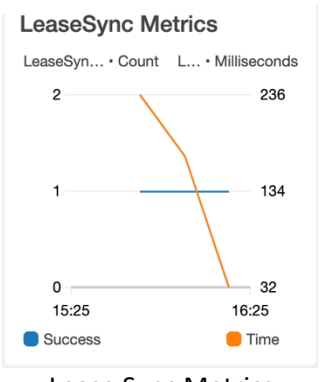
Frame Metrics.



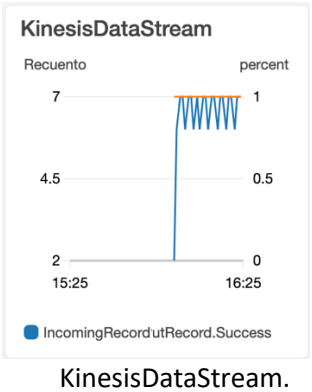
IngestToProcessLatency



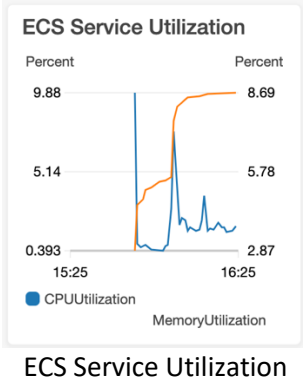
Current Lease Total



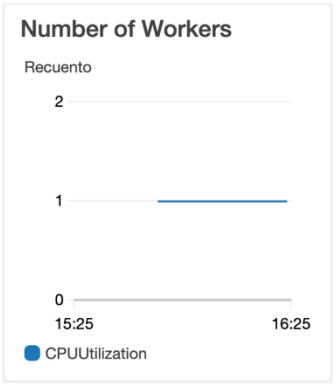
Lease Sync Metrics.

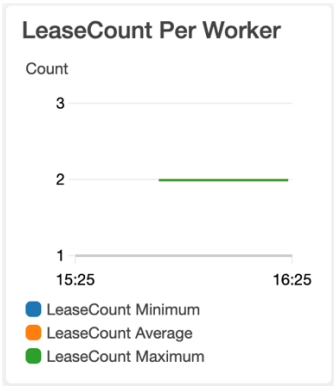


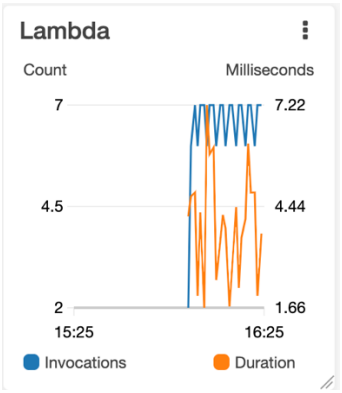
KinesisDataStream.



ECS Service Utilization







Number of Workers LeaseCount por Worker. Lambda

Figura 21: Dashboard CloudWatch resultados

- Fotograma Metrics: métricas para procesar la transmisión de vídeo, enviar fotogramas al *endpoint* de Amazon SageMaker y la escritura de la salida de información de esta a la función Lambda por medio de Kinesis Data Streams.
- IngestToProcessLatency: la diferencia de tiempo entre que se introduce un fotograma en Kinesis Video Streams y el momento que la aplicación lo recibe.
- Current Lease Total: el total de asignaciones actuales de Kinesis Video Streams activas. En nuestro de caso de uso será uno.
- Lease Sync Metrics: metricas de sincronización de transmisiones.
- LeaseCount por Worker: recuento de asignaciones por proceso de trabajo.
- Number of Workers: número de procesos de trabajo.
- ECS Service Utilization: métricas de uso del clúster de Amazon ECS.
- KinesisDataStream: métricas de este servicio.
- SageMaker: operaciones realizadas por el bloc de notas de Amazon SageMaker.
- Lambda: número y duración de la función Lambda que procesa la salida del bloc de notas de Amazon SageMaker.

8- Costes

Aquí analizaremos los costes de las diferentes implementaciones

8.1- Costes Serverless Solution for Video Fotograma Analysis and Alerting

8.1.1 Caso de uso:

El escenario sería el portátil está durante 8 horas analizando si quien entra de una habitación porta un arma. La cámara enviará un fotograma por segundo. En caso de que alguien entre con una pistola se enviará un sms a la persona que se haya indicado. El coste se calculará por día y mes.

8.1.2 Estructura que desplegar:

- Cámara portátil FaceTime HD del portátil
- Amazon Kinesis Data Stream
- Amazon Lambda Function
- Amazon Rekognition Image
- Amazon SNS
- Amazon S3
- Amazon DynamoDB
- Amazon API Gateway

8.1.2 Precios (Irlanda):

- Amazon Kinesis Data Stream [31]: Hora de fragmento (1 MB/segundo de entrada, 2 MB/segundo de salida) 0.017 USD
- Amazon Lambda [32]: capa gratuita incluso después de 12 meses es de 1000000 peticiones. Suficiente para el escenario planteado.
- Amazon Rekognition Image [33]: 1 USD por cada 1000 imágenes procesadas. Por el primer millón de imágenes procesadas al mes. Después del millón desciende 15 céntimos.
- Amazon SNS: el primer millón de solicitudes SNS es gratuito y cada SMS a España y la compañía Vodafone es de 0.08802 USD [34]
- Amazon S3 [35]: Los primeros 50 TB/mes son 0.023 USD por GB
- Amazon DynamoDB [36]:
    - Por unidades de solicitud escritura: 1.4135 USD por millón
    - Por unidades de solicitud de lectura: 0.283 USD por millón
- Amazon API Gateway [37]: 3.5 USD por millón de solicitudes

8.1.3 Calculo Costes:

- Coste 1 día 8 horas de uso:
    - Amazon Kinesis Data Stream: 8x0.017 = **0.136 USD/día**
    - Amazon Rekognition Image: 8horas son 22880 segundos e imágenes procesadas, al no superar el millón queda en **22.88 USD**.
    - Amazon SNS: en el caso poco probable de que entre alguien con un arma nunca excedería el millón de peticiones de SNS y si suponemos que entra una persona durante el día armada el coste del sms es de **0.08802 USD**.
    - Amazon S3: 22880 MB son 22.88 GB esto son **0.52624 USD**
    - Amazon DynamoDB: 1.4135 + 0.283 = **1.6965 USD**
    - Amazon API Gateway: **3.5 USD**
    - Total: **28.74 USD**

- Coste 30 días 8 horas por día:

- Al producirse más de 1 Millón de imágenes al mes esto hace que Rekognition, DynamoDB y API Gateway dupliquen el precio.
- Total: **862.4 USD/mes**

8.2- Costes Amazon Kinesis video Streams Face Detection:

8.2.1 Caso de uso:

Suponemos el siguiente escenario como caso de uso de la arquitectura probada el detectar quien entra y quien sale de una habitación con el portátil encendido durante 8 horas al día con la aplicación encendida y preparada para detectar caras.

8.2.2 Estructura que desplegar:

- La cámara FaceTime HD del portátil transmite a 1 Mbps.
- Amazon Kinesis Video Stream
- Amazon Rekognition Video

8.2.3 Precios:

- El precio por incorporar video a Amazon Kinesis Video Streams por GB en la región de Irlanda es 0.00944 USD. [38]
- el precio por minuto analizado de *streaming* por Rekognition en Irlanda es de 0.12 USD. [33]

Como parte de la capa de uso gratuita de AWS, los clientes nuevos de Amazon Rekognition Video pueden analizar 1 000 minutos de video. Aunque esto solo será durante los 12 primeros meses y en los cálculos no lo vamos a contemplar.

8.2.4 Cálculo de precios:

Precio por día 8 horas son 28800 segundos esto son 28800 Mbits, 3600 MB o 3.515625 GB de transmisión de *streaming* de video a 0.00944 USD por GB da **0.0331875 USD/día**.
A esto hay que sumarle el análisis de Rekognition a 0,12 USD por minuto de análisis teniendo 8 horas cada día da **57,6 USD/día**.
Total: 0.0331875 + 57.6 = **57.6331875 USD/día**
Vemos que donde está realmente el mayor coste es en Amazon Rekognition.
En el caso de querer tener activo esta solución al mes si contamos con que estaría operativo 30 días ascendería a **1728.995625 USD**.

8.3- Costes análisis de videos almacenados

En el caso probado de esta solución se uso un video de un tráiler de la película Casino Royal y un fragmento de un partido de baloncesto. Estas dos pruebas se presentan escasas para analizar las posibilidades que tiene Amazon Rekognition.

8.3.1 Caso de uso:

Para analizar costes de un mes pondrá el siguiente escenario. Analizar todos los partidos de futbol que se celebran en un mes normal de liga de futbol usando Amazon Rekognition Person Tracking.

- 20 equipos en primera división
- Cada semana 10 partidos
- 40 partidos por mes
- A 95 min por partido de media da 3800 minutos de video al mes.
- Se establece video a 4K 20mbps de *bit rate* 556.6 GB aproximadamente en video.

8.3.2 Estructura que desplegar:

- Amazon S3 donde almacenar los videos
- Amazon Rekognition Video
- Amazon SNS
- Amazon SQS

8.3.3 Precios:

- Amazon S3: Los primeros 50 TB/mes son 0.023 USD por GB
- Amazon Rekognition en video almacenado: 0.10 USD por minuto analizado.
- Amazon SNS: el primer millón de solicitudes SNS es gratuito
- Amazon SQS: el primer millón de solicitudes SQS es gratuito

Cálculo de precio:

- Por almacenamiento: 556.6 x 0.023 = 12,802734375 USD/mes
- Por el análisis: 3800 x 0.10 = 380 USD/mes
- Total: 392,802734375 USD/mes

8.4- Costes Análisis video Stream con SageMaker

8.4.1 Caso de uso:

Se quiere hacer un caso parecido al de detectar quien entra y sale de una habitación, pero en este caso será si durante las 8 horas que está conectado la solución se detectan.

- Móvil
- Un ratón de ordenador
- Un libro
- Un monitor/TV

Por seleccionar algunas de las categorías que puede detectar la solución construida.
Además también calcular el precio del caso de uso de 6 cámaras grabando a 4k pero sólo 2 para el análisis.

8.4.2 Capa gratuita de Amazon SageMaker:

- 250 horas para instancias de notebook t2.medium al mes para crear modelos
- 50 horas en instancias de entrenamiento m4.xlarge
- 125 horas de instancias m4.xlarge para implementar modelos de aprendizaje automático. Establecer *endpoint*.

Para establecer los precios no se usarán las horas gratuitas que ofrece la capa gratuita durante los 12 primeros meses después de crear una cuenta.

8.4.3 Instancias usadas para este caso de uso:

- Para ejecutar el notebook: ml.t2.medium
- Para el entrenamiento: ml.p3.2xlarge
- Para el *endpoint*: ml.m4.xlarge

8.4.4 Precios de las instancias (Irlanda) [39]:

- ml.t2.medium: 0.05 USD/hora
- ml.p3.2xlarge: 4.627 USD/hora
- ml.m4.xlarge: 0.311 USD/hora

8.4.5 Calculo de precios:

- Ejecutar todo lo necesario de object_detection_image_json_format.ipynb en la instancia notebook ml.t2.medium nos llevará algo más de una hora. Pondremos 2 horas para redondear.
2 horas de ml.t2.medium dan 0.1 USD
- Para completar el entrenamiento se necesitan 56 minutos en total. Redondeando a 1 hora.
1 hora de ml.p3.2xlarge son 4.627 USD
Una vez montado todo lo anterior ya solo falta poner en marcha la detección:

- En el caso del *endpoint* (ml.m4.xlarge):
8 horas x 30 días x 0.311/hora = 74,64 USD/mes
- Amazon Kinesis Video Streams [38]:
La misma situación que en el problema anterior precio por día 8 horas son 28800 segundos esto son 28800 Mbits, 3600 MB o 3.515625 GB de transmisión de streaming de video a 0.00944 USD por GB da **0.0331875 USD/día**. Al mes **0,995625 USD/mes**.
- Total:
Instancia para el *endpoint* + Amazon Kinesis Video Stream = **75,635625 USD/mes**

- Caso de uso de las 2 cámaras: AKVS 2 horas son 7200 segundos a 100 Mbps cada camara (el maximo que permite AKVS) son 720000 Mbits, son 90 GB a 0.00944 USD/GB = 0.8496 USD.
Instancia 2 horas de ml.m4.xlarge son 0.622 USD. Total = **1.4716 USD**

8.5- Costes DeepStream EC2

8.5.1 Caso de uso:
Caso de uso en el que usamos una instancia EC2 con DeepStream Instalado donde se usará para analizar video como alternativa a Rekognition. Este caso de uso seria con las dos cámaras 4K para la solución que queremos montar grabando durante 2 horas.

8.5.2 Instancia usada:
Usaríamos la instancia más barata la p3.2xlarge con precio de 3.305 dólares.

8.5.3 Calculo precios:

- 2 streams Kinesis Video Streams, 7200 segundos 2 horas x 100Mbps son 90 GB a 0.00944 USD/GB son 0.8496 USD x 2 camaras el total es: 1.6992
- 2 horas de instancia son: 6.61 USD
- Total: **8.3092 USD**

The table shows current software and infrastructure pricing for services hosted in **EU (Ireland)**. Additional taxes or fees may apply.

NVIDIA Deep Learning AMI

EC2 Instance type	Software/hr	EC2/hr	Total/hr
p3.2xlarge ★*Vendor Recommended*	$0	$3.305	$3.305
p3.8xlarge	$0	$13.22	$13.22
p3.16xlarge	$0	$26.44	$26.44
p3dn.24xlarge	$0	$33.711	$33.711

Figura 22: Precio de instancia Nvidia AMI

8.6 Costes uso de Amazon Rekognition Imágenes

8.6.1 Caso de usos:
El caso de uso sería el mismo de antes, 2 cámaras 4K para el análisis, pero esta vez no enviaremos video si no imágenes 15 fps para ser exactos durante dos horas.
También en el caso de 6 fps.

8.6.2 Calculo precios:
- 2 horas son 7200 segundos cada segundo enviamos 15 fotogramas y son dos cámaras esto son en total 216000 fotogramas enviados en las 2 horas a 1 USD por cada 1000 imágenes analizadas por Amazon Rekognition hace un total de **216 USD**.
- si se envían 6 fps el total son **86.4 USD**

9- Conclusiones

Amazon Web Services proporciona un gran abanico de opciones y pone a disposición del desarrollador tecnologías muy potentes para desempeñar la tarea de análisis de video y todo lo necesario para construir estas soluciones.
Tras las pruebas realizadas en las implementaciones podemos ver que el uso de Amazon Rekognition por su sencillez de programación y ser una herramienta muy potente y fácil de desplegar en comparación con el crear tu propio modelo entrenado para el análisis de objetos podría ser la solución a implementar en nuestro caso de uso.
Amazon Rekognition tiene a su vez mucha más versatilidad en cuanto a las diferentes opciones de análisis que proporciona.

- Análisis de objetos, actividades, escenas.
- Análisis de caras tanto en la detección de caras, su posición si está sonriendo, la edad, el género… como en reconocimiento de caras concretas de una colección.
- Análisis del recorrido de las personas.
- Reconocimiento de famosos
- Detección de contenido para adultos.
- Texto en imágenes.

Todo esto siempre en constante mejora, añadiendo nuevas etiquetas y mejorando rendimiento periódicamente.
Pero después de calcular los precios de cada solución y sobre todo las dos posibles soluciones para el escenario de las 6 cámaras podemos ver que el uso de Rekognition Imágenes se presenta como una opción muy cara como para sustituir una configuración *edge* de análisis de video. Pues en dos horas de análisis, capturando 15 fps y usando dos cámaras, el precio asciende a 216 USD. Reduciendo el número de fps a 6 esto reduciría el precio a 86.4 USD siendo aún muy caro. Como opción secundaría la idea de sustituir Rekognition por otro software de análisis de video como DeepStream o otros desplegándolo en una instancia EC2 sería una opción más interesante que da AWS. A la espera de que en el futuro Amazon Rekognition Video pueda analizar videos en *streaming* con Amazon Kinesis Video Streams más allá de la detección de caras. Si comparamos los precios usando una instancia con DeepStream y una tarjeta grafica de Nvidia nos da 8.3 USD por las dos horas y si usamos un modelo propio de entrenamiento en un *endpoint* de SageMaker donde no se necesita que tenga tarjeta gráfica el precio desciende hasta 1.47 USD por las dos horas. Siendo esta última la mejor de las opciones en cuanto a precio.

La conclusión sería que para nuestro caso de uso particular de las 6 cámaras no sería viable hacer una migración a la nube si se utiliza Amazon Rekognition, pues es mucho más cara a la larga que una solución en el *edge*. En cambio, el uso de un *endpoint* de SageMaker con un modelo entrenado si sería viable hacer la migración a esta solución para el caso de uso de este trabajo.
Todo esto sin mencionar lo importante que es tener una buena conexión a internet para proporcionar un flujo constante de fotogramas a AWS en nuestro cliente y los costes que representa y que no se han tomado en cuenta en este trabajo.

Se proporciona una posible solución y su esquema para el problema de la estructura de 6 cámaras en 4K 60 fps en un caso de uso real en la industria:

Dado que para tener 6 cámaras a 4K a 60 fps se necesitaría que el cliente tuviera una conexión a la nube de 900 Mbps y que cada cámara tuviera un *stream* con capacidad de 150 Mbps siendo el límite de Kinesis Video Streams de 100 Mbps. Se necesita una alternativa para poder

llevar a cabo esta tarea. Además de que de poder hacerse sería muy caro llevar esas 6 cámaras a AWS. Para ello usaremos una estrategia hibrida en la que solo usaremos 2 cámaras (máster) para analizar el video. En un evento serían las que enfocan a toda la superficie.
De esas dos cámaras enviaremos 1 de cada 10 fotogramas (6 fps) directamente a Rekognition no usando Amazon Kinesis Video Streams usando la API DetectLabels y quedándonos con la etiqueta *Person* y su *BoundingBox*.
También se contempla en el esquema el uso del *endpoint* con una solución de detección de objetos y personas con SageMaker.
Las respuestas generadas tanto por Rekognition como por la instancia EC2 de SageMaker se devuelven al *edge* y este se encargará de administrar esta información para colocar las demás cámaras de forma adecuada. Además, el propio *edge* se encargará de transmitir el resultado vía Youtube.
A continuación, el esquema de esta solución:

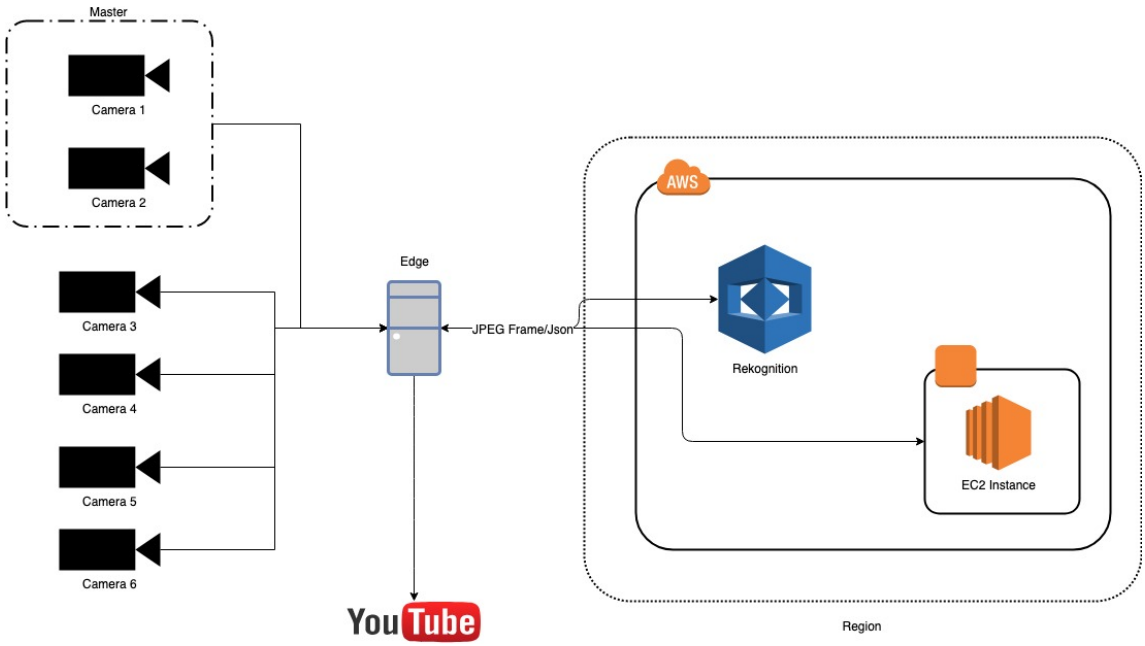


Figura 23: Esquema de la solución llegada en las conclusiones

10- Trabajo Futuro

Implementación y rendimiento:
Como trabajo futuro quedaría la implementación de la solución anterior, comprobar su rendimiento y ver si con 6 fps es suficiente para poder llevar a cabo la tarea o incluso si se puede reducir este. Y la implementación de la versión de DeepStream y ver su funcionamiento.

Implementaciones SageMaker:
Una vez visto lo caro que resulta el análisis de video con Rekognition Imágenes cuando se quiere analizar un video en directo, y la limitación que supone que las diversas funciones de análisis de video de Rekognition Video estén únicamente para videos almacenados, explorar otras opciones de detección de objetos, caras o otras cosas con modelos en SageMaker.

Otros usos de esta misma arquitectura:
Se podría usar esta misma idea como solución para un problema de seguridad en la entrada de un edificio durante la noche. Las dos cámaras máster apuntando con un ángulo amplio de visón que abarque toda la parte exterior del edificio. Enviar 1 de cada 4 fotogramas a Rekognition con la intención de detectar personas. En el caso de detectar una persona activar la cámara que mejor este posicionada para visualizar mejor a esa persona y activar un stream procesor para detección de caras en una colección y ver si esa persona es personal autorizado o no. Durante el tiempo que se detecte la persona, se desplegaría Amazon Kinesis Video Stream pues para la detección de caras en una colección necesitamos crear un stream processor con uno y con un Kinesis Data Stream para obtener los resultados.
En caso de no ser una persona reconocida se activaría una alarma con SNS para enviar un sms al personal de seguridad. Si la persona es personal autorizado el stream processor se desconectaría para no pagar más.
También se le podría añadir el reconocimiento de objetos para saber si va armado.

11- Referencias

[1] Amazon Kinesis Data Streams: https://aws.amazon.com/es/kinesis/data-streams/

[2] Amazon Kinesis Video Strams: https://aws.amazon.com/es/kinesis/video-streams/

[3] Limite Amazon Kinesis Video Streams:
https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/limits.html

[4] Información del productor de video proporcionado por Amazon:
https://docs.aws.amazon.com/es_es/kinesisvideostreams/latest/dg/producer-sdk-cpp.html

[5] Amazon Rekognition: https://aws.amazon.com/es/rekognition/

[6] Diferentes funcionalidades Rekognition:
https://docs.aws.amazon.com/es_es/rekognition/latest/dg/how-it-works-types.html

[7] Detección de rostros en una imagen:
https://docs.aws.amazon.com/es_es/rekognition/latest/dg/faces-detect-images.html

[8] Amazon EC2: https://aws.amazon.com/es/ec2/

[9] Amazon Lambda: https://aws.amazon.com/es/lambda/features/

[10] Amazon SNS: https://aws.amazon.com/es/sns/

[11] Amazon SQS: https://aws.amazon.com/es/sqs/

[12] Diferencias entre SNS y SQS: https://aws.amazon.com/es/sqs/faqs/

[13] Amazon S3: https://aws.amazon.com/es/s3/

[14] Amazon DynamoDB: https://aws.amazon.com/es/dynamodb/

[15] Amazon API Gateway: https://aws.amazon.com/es/api-gateway/

[16] Amazon CloudFormation: https://aws.amazon.com/es/cloudformation/

[17] Amazon Elemental MediaLive: https://aws.amazon.com/es/medialive/

[18] Amazon SageMaker: https://aws.amazon.com/es/sagemaker/

[19] Nvidia DeepStream: https://developer.nvidia.com/deepstream-sdk

[20] AWS Command Line Interface: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

[21] Librerias Boto3 python: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html?id=docs_gateway

[22] Funcionalidades gratuitas AWS: https://aws.amazon.com/free/
[23] Amazon ejemplo serverless, Autor: Moataz Anany https://aws.amazon.com/es/blogs/machine-learning/create-a-serverless-solution-for-video-fotograma-analysis-and-alerting/

[24] Video del tráiler de 007: https://youtu.be/36mnx8dBbGE

[25] Video testeado de Baloncesto: https://youtu.be/Lo9QaAA_y04

[26] Ejemplo de Detección de objetos usando SageMaker: https://aws.amazon.com/es/blogs/machine-learning/analyze-live-video-at-scale-in-real-time-using-amazon-kinesis-video-streams-and-amazon-sagemaker/

[27] Parametros de SageMaker: https://sagemaker.readthedocs.io/en/stable/estimators.html

[28] Hyperparametros de SageMaker para detección de objetos: https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection-api-config.html

[29] Conexión ssh con instancias EC2: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html

[30] API GetMedia: https://docs.aws.amazon.com/es_es/kinesisvideostreams/latest/dg/API_dataplane_GetMedia.html

[31] Precio Kinesis Data Stream: https://aws.amazon.com/es/kinesis/data-streams/pricing/

[32] Precio de Amazon Lambda https://aws.amazon.com/es/lambda/pricing/

[33] Precios Rekognition: https://aws.amazon.com/es/rekognition/pricing/

[34] Precio de SNS: https://aws.amazon.com/es/sns/pricing/

[35] Precio Amazon S3: https://aws.amazon.com/es/s3/pricing/

[36] DynamoDB precios: https://aws.amazon.com/es/dynamodb/pricing/

[37] Amazon API Gateway precios: https://aws.amazon.com/es/api-gateway/pricing/

[38] Precios Kinesis Video Streams: https://aws.amazon.com/es/kinesis/video-streams/pricing/

[39] Precio SageMaker: https://aws.amazon.com/es/sagemaker/pricing/

[40] Regiones donde están alojados AWS: https://aws.amazon.com/es/about-aws/global-infrastructure/regional-product-services/

[41] Amazon IAM: https://aws.amazon.com/es/iam/

[42] Data set COCO usado para el entrenamiento: http://cocodataset.org/#home

[43] SSD algoritmo https://arxiv.org/abs/1512.02325

[44] Ejemplo de FaceDetection. Autor Young Yang: https://github.com/imyoungyang/video-streaming

[45] Haar Cascade Face Detection: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html